情報可視化手法記述のためのデータフロービジュアル言語を備えた開発環境に関する研究

情報可視化手法記述のための
データフロービジュアル言語を備えた
開発環境に関する研究

２０１７年　３月

伊藤　隆朗

情報可視化手法記述のための
データフロービジュアル言語を備えた
開発環境に関する研究

伊藤　隆朗

システム情報工学研究科
筑波大学

2017年3月

概要

情報可視化手法とは，情報可視化におけるデータの視覚的な表現方法のことである．情報可視化を有効活用するためには，導入先の実システムに応じて適切な情報可視化手法を設計する必要がある．

情報可視化手法は，手法の選択肢が広いうえに，適切な手法の範囲が狭い．さらに，設計した手法が妥当か否かの検証には，その可視化手法のための可視化処理の実装が必要となる．そのため，設計者には可視化処理の実装が繰り返し求められる．その対策として，開発現場では，実装が容易な環境下でプロトタイプを制作し，手法の妥当性が確認できた後に，実システムに再実装することがよく行われる．しかしながら，そのような実装の手間の多さは情報可視化の利用を妨げる一因と考えられる．本研究では，情報可視化の利用促進のため，その妨げとなる実装の手間を軽減することを目的に，プロトタイプ制作と実システムへの実装の両方を支援する開発環境を構築した．

プロトタイプの制作作業の本質は，可視化手法をプログラムとして記述することにある．手法の記述に対して，これまでにビジュアルインタフェースを用いる利点が示されてきた．本研究では，その利点を継承しつつ，さらに，従来形式のユーザインタフェースがデータの流れを開発者に提示していないという点に着目し，情報可視化手法の記述のためのデータフロービジュアル言語と，その言語を用いたユーザインタフェースを開発した．データフロービジュアル言語には，多様な情報可視化手法を簡潔に記述できるように，データを視覚的な値へと変換する変換方法と視覚的な値をもとに図形を描くマークを構築部品として用意した．開発したユーザインタフェースが，使いやすさを向上させたことを，被験者実験を通して確認した．また，可視化手法の記述例を通して，多様な可視化手法を簡潔に記述可能なことを示した．

実システムへの実装では，従来，プロトタイプ環境で実装した処理を，実システムで用いられているプログラミング言語や描画 API 等に合わせて再実装する手間が発生することがあった．その手間を軽減するために，様々な環境を考慮した埋め込み可能な実行環境を設計した．組み込み用スクリプト言語による実行環境を設計し，その実行環境を埋め込む際の手間を少なくするようなインタフェース関数を設計した．そして，構築した開発環境に，その組み込み用スクリプト言語のソースコードを出力する機能を加え，様々な実システムの開発において，構築した開発環境を利用できるようにした．

構築した開発環境の実システム開発への利用可能性を検証するために，実際の情報可視化に関する研究に利用した．その研究は，可視化手法の設計を支援するために，時間軸を用いた量的データの可視化手法を提案し，様々なバリエーションの手法を実験ツールに実装して，読み取りに関する比較調査を行うものであった．構築した開発環境を利用し，実験ツールの作成を支援できることを確かめた．さらに，その研究の結果，時間軸を用いることで，量的データを提示するための視覚的属性の種類を増やせることが分かった．

目 次

図 目 次

第1章　序論

本論文では,「情報可視化手法記述のためのデータフロービジュアル言語を備えた開発環境」の構築に関して行った一連の研究について述べる.「情報可視化」とは，データに対する人間の理解を促すことを目的にデータを視覚的に表現することである．また,「情報可視化手法」とは，情報可視化におけるデータの視覚的な表現方法のことである.「データフロービジュアル言語」とは，データの流れを視覚的要素の組み合わせによって記述する言語のことである．本章では，まず，本研究の意義について説明したのち，課題を明確にする．次に，設定した課題に対する成果について説明する．最後に，本論文の構成を示す.

1.1　本研究の意義

情報可視化は，データの検索，モニタリング，分析など，様々な場面で利用されている．近年のデータ量の増加から，情報可視化を自らのシステムにも利用したいという需要は高く，今後ますます発展が期待される分野だと考えられる.

情報可視化の重要な役割は，対象データや利用目的に応じてデータを効果的に提示することである．実際のシステム（以下,「実システム」と呼ぶ）において情報可視化を有効活用するためには，そのシステムのデータや利用目的に応じた適切な可視化手法を設計することが重要となっている．可視化手法を設計する際，全く新しい手法を考案することは必ずしも必要ではないものの，既存の可視化手法をデータを適切に表現できるようにカスタマイズすることが必要になることが多い [HCL05]．しかし，可視化手法の設計には，次のような難しさがある．まず，情報可視化では，成績や売上数のような実空間に座標が存在しないデータを取り扱うため，用いられる可視化手法が多様である．その多様な手法の中から利用目的やデータに応じた適切な手法を見つけだす必要があるが，適切な手法の範囲は狭いことが多く [Mun14]，このことが難しさの一つである．次に，設計している可視化手法がデータを効果的に表現できるかどうかについては，その可視化手法による結果を得るための可視化処理をプログラムに実装し，データを可視化してみるまで検証や判断が困難という，もう一つの難しさがある．まとめると，可視化手法を設計する際，可視化手法の設計の範囲が広いうえに，適切な手法の範囲は狭く，かつ，設計した手法が妥当か否かの検証には，その可視化手法のための可視化処理の実装が必要であることが，設計の難しさといえる．このような難しさから，実システムの開発では，可視化処理の実装が容易な環境下でプロトタイプを制作し，その後，実システムに再実装するプロセスがとられることが多い.

本研究では，実システムにおける開発プロセスを支援するための開発環境を構築し，その

開発環境を Iv Studio と名付けた．Iv Studio には，情報可視化手法を記述するためのデータフロービジュアル言語がユーザインタフェースに用いられており，開発者は可視化処理を容易に実装でき，ツール上で可視化結果を即座に確認できる．さらに，実システムへの組み込みが可能な情報可視化の実行環境が提供されており，開発者は実装した可視化処理を実システムから容易に利用できる．つまり，Iv Studio は，プロトタイプ制作と実システムへの実装の両方を支援する開発環境である．

プロトタイプ制作と実システムへの実装の両方を支援する開発環境を構築することの意義は 2 つある．

一つ目の意義は，実システムにおける情報可視化の利用の促進につながることである．情報可視化をシステムに利用したいという需要は高いものの，実装のハードルの高さが利用を妨げる一因となっている．このハードルを引き下げることで，情報可視化がさらに多くのシステムで利用されるようになると考えられる．

もう一つの意義は，情報可視化研究の発展を促進することにつながることである．情報可視化研究では，利用目的やデータに応じた適切な可視化手法の探索を容易にするための研究が盛んに行われている．その研究過程では，多くの手法を考案し，実際に検証することが必要となる．つまり，数多くの可視化処理を実装する必要がある．そのため，可視化処理の実装や実験に用いる実システムへの実装を支援することは，そのような情報可視化研究の発展につながると考えられ，ひいては，適切な可視化手法の探索を容易にすることにつながると考えられる．

1.2 本研究の目的と課題

本研究の大きな目的は，情報可視化の利用を促進することである．この目的は，非常に大きなものであり，技術的課題として捉えるには抽象的なことから，より具体的な目的を設定する．本研究では、情報可視化の利用を妨げる一因として，実システムに利用する際に生じる実装の手間の多さに着目する．そして，そのような実装の手間を軽減するために，プロトタイプ制作と実システムへの実装の両方を支援することを目的とする．

現在，情報可視化処理の多くは，テキストベースのプログラミング言語（以下，単に「プログラミング言語」と記す）によって実装されることが多い．一方，近年の研究 [SH14, RHY14] により，ビジュアルインタフェースによる情報可視化処理の実装の有効性が示された．視覚的なパラメータを直感的に調整できることや，可視化結果を即座に確認できるという利点は，設計中の手法の検証を即座に行えるため，特にプロトタイプ制作において有効である．筆者は，可視化処理の実装において，ビジュアルインタフェースを用いた開発ツール（以下,「ビジュアルツール」と呼ぶ）が，プログラミング言語に代わるものになると考えている．そのため，本研究では，ビジュアルインタフェースを備えた情報可視化処理のためのプログラムの開発環境を提供するというアプローチをとる．

しかしながら，既存のビジュアルツールには以下の課題が残されている．

課題 1 : ユーザインタフェースの使いやすさ

可視化処理のプログラムの作成においては，当然ながら可視化処理の手続きをプログラムとして記述する作業が行われるが，その本質は，コンピュータが処理可能なように視覚的な表現方法を記述すること，すなわち，可視化手法を記述することにある．可視化手法の記述には，大きく分けて 2 つの記述が必要である．一つ目は，データ，データを表すための視覚的属性（位置や色など），データを視覚的属性に変換するための変換方法（軸，カラーマッパー）の，3 つの対応関係の記述である．もう一つは，データを視覚的属性に変換するために基準となる位置や色といった視覚的パラメータの設定の記述である．後者については，従来のビジュアルツールにおいて十分な支援がなされている [SH14, RHY14]．しかし，前者については，ユーザインタフェースの使いやすさについて改善の余地があると考えている．例えば，既存ツールの利用者は，視覚的属性とデータを対応付ける際に，その視覚的属性を持つマークを選択し，パネルの表示内容を切り替えながら記述していく必要がある．このようなパネル内容の切り替え作業は，ツールの利用者が本来行いたい可視化手法の記述作業の妨げになると考えられる．

課題 2 : プログラムからの可視化処理の利用方法

既存のビジュアルツールでは，実装した可視化処理の実行はツール上か，ツールが提供する実行ランタイム上に限られている．そのため，例えば，既存のプログラムに可視化の機能を追加するような状況，あるいは，システムの要件により開発に用いることのできる環境が制約されているような状況では，ツールの提供するランタイムが対応していない場合，ビジュアルツール上で実装した処理を，既存のプログラムに再実装する必要がある．再実装にかかるコストを削減するためには，情報可視化の利用先のプログラムに用いられている様々な環境から，実装した処理を容易に利用できるようにすることが必要である．

1.3 本研究の成果

1.1 節で述べた通り，本研究の意義は，情報可視化の利用や可視化研究発展の促進である．しかし，どの程度の支援がそれらの促進につながるのかを科学的に解明することは難しい問題であり，今後も多大な研究努力が必要である．現段階における，筆者の主張する本研究の成果は以下のとおりである．

情報可視化手法記述のためのデータフロービジュアル言語を用いたユーザインタフェース

プログラミング言語において，可視化手法の構成要素を組み合わせる記述パラダイムにより様々な手法を簡潔に記述できることが示されてきた．本研究では，その記述パラダイムに則ったデータフロービジュアル言語を開発し，それをユーザインタフェースに採用した．データの流れを開発者に提示するという特徴により，既存のビジュアルインタフェースの利点を継承しながらも，ユーザインタフェースの使いやすさの向上を図った．データの流れを提示しない従来形式のユーザインタフェースとの比較実験を行い，開発したユーザインタフェー

スが，ユーザインタフェースの理解のしやすさと，記述の効率性を向上させたことを示した．

この研究の成果は，可視化手法の構成要素を組み合わせる記述パラダイムにおいて，使いやすさを向上させたユーザインタフェースを開発したことであり，これは，今後，情報可視化手法記述のためのビジュアルインタフェースを開発するうえでの有用な知見である．

実システムへの埋め込みが容易な情報可視化の実行環境

既存のビジュアルツールでは，ツール上で実装した可視化処理の利用先には大きな制約があった．それに対し本研究では，様々なデータ形式，描画 API，プログラミング言語を考慮した，プログラムへの埋め込みが容易な可視化の実行環境を設計し，ツール上で実装した可視化処理の利用先を広げた．

埋め込み可能な情報可視化の実行環境という考え方は，ビジュアルツールの利用先を大幅に広げるものであり，Iv Studio に限らず，今後，情報可視化手法記述のためのビジュアルツールを開発するための有用な指針となる．

時間軸を用いた量的データの可視化手法に関する研究への利用

Iv Studio を，実際の情報可視化研究に利用し，実システム開発への利用可能性を検証した．その研究は，時間軸を用いた量的データの可視化手法に関する研究であり，情報可視化研究において重要な，人の視覚的特性に関する新たな知見を得るためのものである．そのような研究では，数多くの可視化手法を設計しプログラムに実装する必要がある．その部分を Iv Studio は支援することができた．

時間軸を用いた量的データの可視化手法に関する研究への利用は，情報可視化に関する研究を支援した事例の一つであり，ビジュアルツールの利用方法に関する知見を提供するものである．また，その研究の結果，提案した可視化手法によって視覚的属性の種類を増やすことができることが分かり，情報可視化研究に貢献した．

1.4 本論文の構成

次の第 2 章では，構築する開発環境に対する要求と，構築における目標について述べる．第 3 章では，関連研究を紹介し，目標達成の観点から，本研究と従来の研究や既存ツールとを比較する．第 4 章では，情報可視化手法記述のためのデータフロービジュアル言語とユーザインタフェースについて述べ，開発したユーザインタフェースと従来形式のユーザインタフェースとの比較実験の結果とともに，本研究のユーザインタフェースの有効性を示す．第 5 章では，実システムへの埋め込みが容易な情報可視化の実行環境の設計について述べる．第 6 章では，時間軸を用いた量的データの可視化手法に関する研究について述べる．最後に，第 7 章にて結論を述べる．

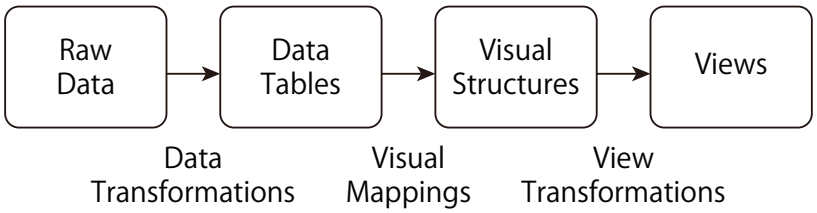
第2章　開発環境の構築における目標設定

本章では，構築する開発環境の対象とする範囲と，目標について述べる．まず，開発環境が支援する対象の範囲について説明する．次に，可視化手法の設計における難しさと，構築する開発環境の目的を説明する．そして，開発環境に対する要求定義と，構築における目標について述べる．

2.1　開発環境が対象とする範囲

まず，構築する開発環境が対象とする，情報可視化（Information Visualization）について説明する．情報可視化とは，人間のデータ理解を支援するために，コンピュータグラフィックスを活用して抽象的なデータを人間が直接見ることができる形に表現することである．情報可視化は，データの検索，モニタリング，分析など，様々な場面で活用されており，近年のデータ量の増加やデータの複雑化から，今後ますます発展が望まれている分野だと考えられる．コンピュータサイエンスにおける可視化の研究には，科学技術計算の結果をコンピュータグラフィックスで表現する科学的可視化 (Scientific Visualization) と呼ばれる研究分野がある．科学技術計算の計算結果は膨大であり，その膨大なデータを処理するためのシステムや演算方法が主な研究対象である．科学的可視化では，実世界の 2 次元/3 次元の対象を取り扱っているため，利用する視覚的な表現方法については，計算対象の座標をそのまま拡大縮小して表示したものを利用すればよい [Fry08]．それに対して，情報可視化では，成績や売上数などの実空間に座標が存在しないデータを取り扱うため，用いられる視覚的な表現方法が科学的可視化の表現方法と比べて多様である．その多様な表現方法の中から，利用目的やデータに応じた適切な方法を見つけだすことが，情報可視化の研究対象のひとつである．以降，単に可視化と記した場合，情報可視化を指すものとする．

情報可視化の処理では大きく，Data Transformations，Visual Mappings，View Transformations の 3 段階の変換処理が行われることが知られている．図 2.1 に，その過程をモデル化した，情報可視化のリファレンスモデル [CMS99] を示す．Data Transformations 処理において，SQL データベース等の生の状態のデータが，可視化用に整理された表の形式へと変換され，Visual Mappings 処理において，表の形式のデータが視覚的な要素であるマークやマークの視覚的属性，マークの位置を司る空間的基板から構成される Visual Structures へと変換される．空間的基板とは，例えば，マークの位置の基準となる軸のことをさす．最後に，View Transformations 処理において，Visual Structures に対してパンやズーミングなどの処理が行われて Views へと変換され，コンピュータのディスプレイに画像が表示される．



Raw Data : 生データの状態（例 : SQL データベース、スプレッドシート等）
Data Tables : 可視化用に加工された表の集合
Visual Structures : 空間的基板 + マーク + マークの視覚的属性の集合
Views : 視覚的パラメータの集合（例 : 位置、大きさ等）

図 2.1: 情報可視化のリファレンスモデル

構築する開発環境では，リファレンスモデルに示された変換処理のうち，データを視覚的な状態へと変換する Visual Mappings 処理の開発支援を主な対象とする．なお，近年の情報可視化では，生データの段階で見ると，数 100 万レコードを超える大規模データや，数十次元を超える多次元データを扱うこともある．このようなデータの可視化では統計処理やフィルタリング，次元圧縮，クラスタリングなどの技術によってデータを削減することが多いが，このような処理は，Visual Mappings 以前の Data Transformations における処理であり，構築する開発環境の対象外とする．

2.2 情報可視化手法

Visual Mappings 処理のプログラム制作においては，データをどのように視覚的に表現するかという表現方法をコンピュータが処理できるようにプログラムに記述することが主要なタスクである．本研究では，情報可視化におけるデータの視覚的な表現方法を「情報可視化手法」と呼ぶ．以降，単に可視化手法と記した場合，情報可視化手法を指すものとする．

情報可視化手法はどのデータを，どのような変換方法を用いて，どのようなマークのどの視覚的属性を用いて表すかという変換規則から構成されるものと考えることができる [Mac86, Mun14]．このことについて，例をあげて説明する．図 2.2(a) は，396 車種の自動車の燃費と馬力の関係を表した図である．この図を描くための手法は，一般的に散布図と呼ばれている．この手法を，先の変換規則の集合として書くと，以下の通りになる．

- 燃費のデータを，X 軸に射影して，円の X 座標で表す
- 馬力のデータを，Y 軸に射影して，円の Y 座標で表す

Visual Structures の要素として考えると，X 軸と Y 軸は空間的基板，円はマーク，X 座標，Y 座標は視覚的属性である．

図 2.2(a) に加え，車重を円の色で表した図が図 2.2(b) である．この図を描くための手法を，変換規則の集合として書くと，以下の通りになる．

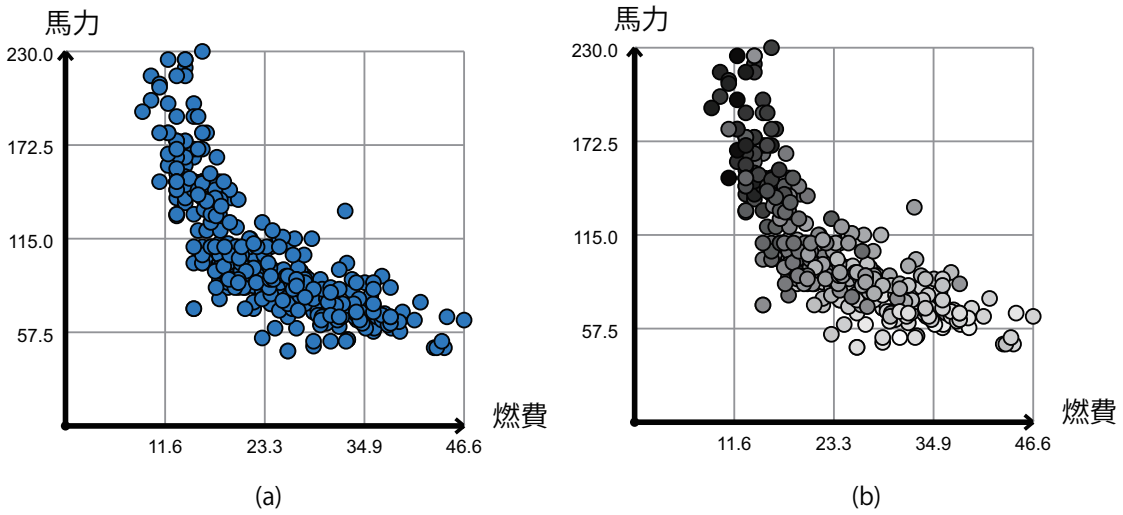


図 2.2: 自動車のデータの可視化例

- 燃費のデータを，X 軸に射影して，円の X 座標で表す
- 馬力のデータを，Y 軸に射影して，円の Y 座標で表す
- 車重のデータを，白から黒の間の色に変換して，円の色で表す

以上のように，可視化手法は変換規則の集合として記述することができる．この考え方は，可視化手法の記述に関する多くの研究や文献 [HCL05, fla, BH09, BOH11, Ber84, Mun14, Mac86] で採用されているものであり，本研究においてもこの考え方に基づいて，可視化手法を取り扱うこととする．

ところで，例えば，図 2.2(b) に関して，単に散布図と呼ぶのか，散布図のカスタマイズ版の手法と呼ぶのかは意見が分かれると考えられる．また，どこまでカスタマイズをすれば別の手法と呼べるのかについても，個々のカスタマイズ内容によるところが大きい．このような，どこまでが同一の可視化手法であるか否かといった，可視化手法の名前による分類方法については，本研究の議論の対象外であると考える．このような議論は，例えば，Excel のような，可視化手法を選択してデータを可視化するようなシステムを構築する際には，必要な議論であるが，本研究のように可視化手法自体を記述するためのシステムを構築する際には必要なものではない．

先に示した規則には,「X 軸に射影」のようにデータの値を視覚的な値に変換するための規則が必要となる．このような変換に用いる技術のことを可視化手法と呼ぶことがある．この場合の可視化手法という言葉では，手法という言葉が「技法，技術」という意味で用いられている．本研究では，手法という言葉を「表現方法」という意味で用いており,「技法，技術」という意味で用いられる場合と区別するために，先のような変換処理のことを可視化技術と呼ぶ．データから座標を計算する技術や，データから色を計算する技術のことを可視化技術とよび，特に，それぞれレイアウト技術やカラーマッピング技術のように呼ぶ．

2.3 可視化手法の設計における難しさと開発環境の目的

情報可視化の重要な役割は，対象データや利用目的に応じてデータを効果的に提示することである．システムにおいて情報可視化を有効活用するためには，そのシステムのデータや利用目的に応じて適切な手法を設計することが重要となっている．全く新しい可視化手法の設計が必ずしも必要とされるわけではないが，少なくとも，システムのデータや目的に適合するように既存の可視化手法をカスタマイズすることや，複数の手法を組み合わせることが必要となることが多い [HCL05]．

可視化手法の設計には次のような難しさがある．まず，値の表現に用いることのできる視覚的属性の種類 [Ber84, Mun14] が少ないことから，手法の設計者は，データを表現するために，複数のマークを用いる工夫や，空間的基板を用いた工夫 [Cha06]，どの視覚的属性を用いてどのデータを表現するかの組み合わせの工夫などが必要となる．このような工夫が，可視化手法の設計作業に当たるものであるが，手法の設計空間は広大であるうえに，ある目的やデータに適合するような手法の範囲は狭いことが多い [Mun14]．場合によっては，目的に合った手法が存在しないか，まだ発見に至っていないこともある．このことが，可視化手法の設計の難しさの一つである．次に，設計している可視化手法がデータを効果的に表現できるかどうかについては，その手法を用いてデータを可視化してみるまで検証が困難なことも，設計の難しさの一つである．そのため，設計者は，設計している可視化手法による可視化結果を得るための可視化処理のプログラムを，検証のたびに実装する必要がある．まとめると，可視化手法を設計する際，可視化手法の設計空間が広いうえに，適切な手法の範囲は狭く，かつ，設計した可視化手法が妥当か否かの検証にはその手法の実装が必要であることが，設計の難しさといえる．このような難しさから，実システムの開発では，可視化処理の実装が容易な環境下でプロトタイプを制作し，その後，実システムに再実装するプロセスがとられることが多い．

可視化手法の設計の難しさに対して，データを表現するための工夫に関する研究や，手法を考案する際のガイドラインに関する研究が数多くなされている．そのような研究の中でも，可視化手法のプロトタイププログラムの実装と修正は繰り返し行われる．また，複数の手法を比較検討するような研究では，研究者は，比較対象の可視化手法を全てを実験用プログラム上で動作するように実装する必要がある．つまり，情報可視化研究においても，プロトタイプ制作と運用するシステムへの実装が数多く行われている．

前章で述べたように，本研究の大きな目的は，情報可視化の利用を促進することである．情報可視化の利用を妨げる一因として，実システムに利用する際に生じる実装の手間の多さに着目する．そして，そのような実装の手間を軽減するために，プロトタイプ制作と実システムへの実装の両方を支援することを目的とした開発環境を構築する．実装の手間の軽減は，可視化手法の設計空間の探索の難しさを解決するための研究の促進にもつながると考えられる．つまり，実システムにおける情報可視化手法の探索を容易にすることにもつながる．そのため，実装の手間の軽減は，情報可視化の利用を促進するという大きな目的に向けた第一歩と考えられる．

2.4 開発環境に対する要求

開発環境に対する要求を以下のように定義した．

Req1. 様々な可視化手法を記述できる（高い表現力を持つ）

Req2. 可視化手法を容易に記述できる

Req3. 可視化のためのプログラムから，開発環境上で実装した可視化処理を容易に利用できる

可視化処理のプログラムの作成においては，当然ながら可視化処理の手続きをプログラムとして記述する作業が行われるが，その本質は，コンピュータが処理可能なように視覚的な表現方法を記述すること，すなわち，可視化手法を記述することにある．先に，ある目的やデータに適合するような手法の範囲が狭いことを述べたが，このことから，構築する開発環境において，記述可能な手法の範囲を広く持つこと，すなわち，Req1 は重要な要求と考えられる．

プロトタイプ制作を支援するという観点から，Req2 は重要な要求である．開発環境の利用者（以下,「開発者」と呼ぶ）が，頭の中に思い描いている可視化手法を計算機に対して直感的に伝えられるようにすることが求められる．より具体的には，可視化手法の記述にかかる操作量の少なさや，記述に用いるユーザインタフェースの使いやすさが求められる．

実システムへの実装を支援するという観点から，Req3 は重要な要求である．開発環境が対象とする Visual Mappings 処理は，可視化を行うプログラム全体から見ると，一部の処理である．そのため，開発環境上で実装した可視化処理を，別のプログラムから利用できるようにする必要があり，その際の利用方法も容易であることが求められる．

2.5 目標の設定

要求をもとに，開発環境の構築における目標を設定する．Req1 と Req2 はトレードオフの関係にある要求である．このような要求に対し，プログラミング言語のツールキットでは，サポートする範囲を想定して，その範囲を簡潔に記述できるようにし，その一方で，それ以外の範囲に対しては拡張性を持たせることで対応することが多い．そこで，構築する開発環境においても，同様に，ある程度の範囲を想定し，その範囲の手法を簡潔に記述できるようにすることを目指す（目標 1）．ここでいう簡潔にとは，手法の記述にかかる記述量を少なくすることとする．そして，想定した範囲外に対して対応するために，開発環境に拡張性を持たせることを目標とする（目標 2）．可視化手法の記述を容易にする（Req2）ためには，単に簡潔に記述できるようにするほかにも，ユーザインタフェースの使いやすさが重要である．ユーザインタフェースの使いやすさについては，より具体的に，2 つに分けて目標を設定する．可視化手法の記述では，大きく分けて，データを視覚的な表現に変換するための手順の記述（目標 3）と，変換する際に必要となる視覚的なパラメータの指定の記述（目標 4）の 2 つが行われる．そのため，それぞれの記述を行いやすいユーザインタフェースを提供するこ

とを目標する．Req3 に対しては，クライアントアプリケーションのプログラムで用いられる主要な環境を想定し，その環境おいて可視化処理を利用できるようにすることを目標とする（目標 5）．以下，設定した目標の詳細を述べる．

目標 1: 多様な可視化手法を簡潔に記述できるようにする

まず，手法の記述に関して，大前提として，可視化手法の探索プロセスに即した機能要件を設定する．多様な可視化手法を記述できるようにしたい理由は，開発者が，目的やデータに適合するような手法にたどり着けるようにしたいからである．たどり着けるようにするためには，開発者が，可視化結果を観察しながら，表現するデータや表現方法（変換規則）の追加や変更を行っていくという探索プロセス（例えば，文献 [Fry08] における表現と緻密化のプロセス）の支援が必要と考えられる．具体的にそのようなプロセスでは，新たなマークを追加することや，マークの持つ視覚的属性に対して新たなデータを割り当てることが行われる．そのため，マークの追加や，マークの持つ視覚的な属性を直接的に制御できる機能を有することを機能要件とする．

次に，可視化手法の範囲について目標を設定する．可視化手法の範囲を明確に定めることは容易でないが，目標として，可視化手法の多様性の観点から範囲を設定する．多様性として，対象データ，表現空間，表現形式における多様性を想定して既存のビジュアルツール [SH14, RHY14] と同程度の基準を置く．くわえて，可視化を効果的にするためは欠かすことのできないインタラクションも想定する．データの種類は量的データとカテゴリデータのどちらも扱えるものとする．また，データ構造としては，多次元データ，グラフ構造，木構造も取り扱えるものとし，グラフ構造や木構造については，その頂点やエッジに重みのついたものを取り扱えるものとする．表現形式に関しては，出原ら [出原 86] によると，我々が日常的あるいは学術的に利用する図は，その形式面に着目することで，値を位置によって表す座標系，領域を囲うことで関係を表す領域系，いわゆる表の形式で関係をあらわす配列系，要素間を線でつなぐことで関係を表す連結系の 4 つの基本系に分けられる．そこで，表現形式としてはこれら全ての基本系を一通りカバーするものとする．表現空間は，既存のビジュアルツールと同様に 2 次元に限定する．インタラクションについては，マルチプルビューでは必須となる Linking & Brushing[Kei02] の記述と，視覚的混雑を軽減するためによく用いられる選択したレコードのみ視覚要素を表示するような手法の記述を可能にするものとする．以上，最低限，ここで基準とした範囲の可視化手法を，少ない記述量で記述できるようにすることを目標とする．

目標 2: 記述できる可視化手法の範囲を拡張できるようにする

高い表現力（Req1）を備えるためには，拡張性が重要である．拡張が可能であれば，開発者の求める可視化手法が目標 1 にて設定した範囲の外であったとしても対応することができる．また，例えば，今後新たな可視化技術が開発された場合にも，対応が可能となる．

そこで，汎用的な演算処理によって，カラーマッピング技術やレイアウト技術の拡張を行えるようにすることを目標とする．

目標 3: 手法を記述しやすいユーザインタフェースを提供する

情報可視化手法は，2.2 にて述べたように，どのデータを，どのような変換方法を用いて，どのようなマークのどの視覚的属性を用いて表すかの規則の集合として考えることができる．この考え方を，本研究では「ビジュアルエンコーディング」と呼ぶ．ビジュアルエンコーディングは，いくつかの可視化のツールキット [HCL05, fla, BH09, BOH11] の設計の基礎となっている．また，多くの文献 [Ber84, Mun14, Mac86] でこの考え方が解説されている．そのため，ビジュアルエンコーディングは，少なくとも可視化手法を考案するうえで重要なものと考えられる．

構築する開発環境では，開発者の思い描いたビジュアルエンコーディングの方法を記述しやすいユーザインタフェースを目標とする．記述のしやすさとしては，開発者が，記述方法を直感的に理解できることと，ビジュアルエンコーディングの方法を効率的に記述できることの 2 つを想定する．くわえて，可視化手法の設計では，可視化手法のカスタマイズが繰り返されることが多いことから，その記述も行いやすくすることをユーザインタフェースの目標とする．

目標 4: 視覚的なパラメータの指定を記述しやすいユーザインタフェースを提供する

情報可視化手法の構築部品には，データから視覚的な値への変換方法がある．大半の変換方法では，データを，図形の色，大きさ，位置などの視覚値へ変換する際に，対応づける色の範囲，大きさへの写像，座標軸の位置など，いくつかのパラメータが必要となる．そのため，開発者は，これらの指定の記述が必要となる．このような視覚的なパラメータは可視化結果の見やすさに大きく影響するとともに，その効果の確認が欠かせないため，容易に指定および修正ができることが望まれる．そのため，視覚的なパラメータの指定の記述が行いやすいユーザインタフェースを目標とする．

目標 5: 可視化処理を様々なプログラムに容易に埋め込めるようにする

開発環境上で実装した可視化処理を，様々なプログラムに埋め込むことができるようになれば，構築する開発環境を多くの実システムの開発に利用できるようになると考えられる．

埋め込み先のプログラムは，以下のような環境で開発されているものを想定する．プログラミング言語としては，クライアントアプリケーションの開発に利用される主要な言語を想定する．具体的には，C，Java，JavaScript，C#を想定する．また，可視化を行うプログラムには，データの読み込みや生成した画像の表示が必須となり，インタラクション手法を用いた場合は入力の受付も必須である．そのため，埋め込み先のプログラムは，必ず何らかのデータ形式や描画 API，GUI ツールキットに依存している．構築する開発環境では，これらの環境には依存しない形で可視化処理を出力することができるようにし，さらに少ない作業量でその処理を埋め込めるようにすることを目標とする．

第3章　関連研究

本章では，情報可視化プログラムの開発環境に関連する研究を述べる．情報可視化に限らずプログラムの制作にはプログラミング言語が広く利用されている．可視化に焦点を合わせると，ビジュアルインタフェースによって手法を記述するツールもいくつか開発されている．可視化処理をデータの処理フローとして見ると，データフロービジュアル言語による記述方法も関連がある．そこで，テキストベースのプログラミング言語，ビジュアルインタフェース，データフロービジュアル言語の，それぞれの観点について 2 章にて設定した目標と照らし合わせながら関連研究を述べる．加えて，Iv Studio を利用して行った時間軸を用いた量的データの可視化手法の研究の関連研究も述べる．

3.1　テキストベースのプログラミング言語による記述

テキストベースのプログラミング言語の使用を前提として，可視化プログラムの開発を支援する様々な試みがある．

可視化プログラムは，最終的に画像を表示するプログラムである．そのようなプログラムの開発を支援する開発環境やツールキットがある．Processing[Pro] は，ウィンドウ生成のような画像を表示するまでのプログラミングを行う必要がなく，可視化結果を表示するプログラムを短いコードで制作できる開発環境である．また，描画 API もシンプルで短い命令となっており，表示処理も簡潔に記述することができる．Processing と同様に，画像を表示するプログラムを短いコードで簡潔に制作できる C++用のツールキットとして，Siv3D[Siv] がある．

情報可視化のためのプログラムに特化したツールキットも開発されている．InfoVis toolkit[Fek04] は，典型的な可視化手法や可視化プログラムでよく使われる GUI コントロールをクラスとして提供する Java 用のツールキットである．Prefuse[HCL05] と Flare[fla] は，可視化プログラムを制作するための粒度の細かい構築部品を提供するツールキットである．例えば，データを受け取り，棒グラフや散布図のような可視化結果を出力するクラスを提供するのではなく，座標軸やカラーマッパーのような視覚値への変換方法や，円や矩形のような視覚要素といった情報可視化手法の構築部品をクラスとして提供する．Prefuse が Java 用のツールキット，Flare が ActionScript 用のツールキットである．可視化手法の構築部品を提供し，それらを組み合わせることで手法を記述するというパラダイムは，それ以降多くのツールキットで採用された．Protovis[BH09] は，抽象度の高いツールキットと抽象度の低い描画 API との間のギャップを埋めるために開発されたシンプルな視覚要素を提供する JavaScript 用のツールキットである．D3[BOH11] は，Protovis のコンセプトに加え，開発環境や将来的な CSS へ

の互換性を持たせた JavaScript 用のツールキットである．このツールキットは，データ数に応じた DOM の生成と，データ内容に応じた DOM の視覚的な属性の制御の記述を支援することで，開発者が様々な可視化手法を簡潔に記述できるようにしている．つまり，D3 を用いることで，開発者は，データによって DOM を制御するという，データドリブンのパラダイムで，可視化手法を記述することができる．D3 は，多くの Web サイトで利用されており，現在最も普及している可視化向けのツールキットと言える．

プログラム制作で利用するための実装を提供するツールキットとは異なり，プログラムの設計方法に関する研究もなされている．特に，情報可視化の分野に限定すると，可視化システム構築のためのデザインパターン [HA06, GE08] の研究がある．これらの研究では，プログラム上でのデータの持ち方や，可視化の実行システムの構築の仕方が提案されている．

プログラミング言語の利点は，非常に広い範囲の可視化手法を記述できることである．しかし，テキストによる記述はデータと視覚要素の結びつきを理解しづらいという欠点や，位置や配色に関わるパラメータの調整が行いづらいという欠点が，文献 [SH14, RHY14] で指摘されている．くわえて，編集結果を確認するためにコンパイル等の作業が必要である．以上の理由から，目標 3，目標 4 について不十分である．目標 5 については，ツールキット等が，利用先のシステムの環境（プログラミング言語や描画 API）に対応していれば，記述した可視化手法をそのまま利用することができるものの，利用先のシステムと合わない場合は，再実装が必要になる．デザインパターンは目標 5 を達成するにあたり重要な指針になるものの，目標達成のためには，埋め込み先の言語や描画 API を考慮した実行環境の選定のような，より実際的な部分に着目する必要がある．表 3.1 にプログラミング言語を前提とした可視化プログラム開発支援における目標の達成状況を示す．

3.2 ビジュアルインタフェースによる記述

ビジュアルインタフェースによって可視化手法を記述する方法としては，可視化手法を実装した可視化ビューを組み合わせる方法と，可視化手法の構築部品を組み合わせる方法が多く採用されている．可視化ビューとは，データを受け取り，可視化結果を出力するような機能単位のことである．例えば，2 カラムのデータテーブルを受け取り，散布図を表示するような機能は，ひとつの可視化ビューである．

可視化ビューを組み合わせることのできるツールとして，Improvise[Wea04]，Snap-Together Visualization[NS00] が挙げられる．Improvise や Snap-Together Visualization は，予め用意されたビューを互いに連携させることのできるユーザインタフェースを備えている．それを用いて，利用者は，マルチプルビューの可視化手法を用いたデータの分析システムを構築することができる．

可視化手法の構築部品を組み合わせることで，新たな可視化手法の記述や，可視化手法のカスタマイズの記述を可能とするツールも開発されている．FLINA[CvW11] は，利用者が，キャンバス上に座標軸を描き，データを割り当てることで，散布図や PCP を組み合わせたチャートを制作することができるツールである．GLO-STIX[SKL+14] は，グラフ構造の可視化手法

の要素となる 6 つの技術を部品として組み合わせることで，グラフ構造のための可視化手法を記述できるツールである．Lyra[SH14] や iVis Designer[RHY14] は，データ，視覚値へのデータ変換，視覚要素といった可視化手法の構築部品を組み合わせることで，様々なチャートを制作することができるツールである．Lyra や iVis Designer では，ドローツール風インタフェースを用いて，座標軸の位置や視覚要素の大きさ，色をインタラクティブに調整することができる．

複数の可視化ビューを組み合わせる方法では，可視化ビューに対してデータを設定することでデータを容易に可視化することができるものの，利用できる手法は予め用意されたものに限られる．また，可視化ビューで用いられている手法のカスタマイズをビジュアルインタフェースから行えない．例えば，散布図と棒グラフを実装した可視化ビューが用意されていたとすると，その表現を組み合わせたマルチプルビューの可視化手法を構築することはできるものの，散布図の点の色によって値を表すような手法は利用できない．そのため，目標 1 に対して不十分である．

可視化手法の構築部品を組み合わせる方法は，多様な可視化手法を記述することができる．ただし，FLINA に用意されている部品は，軸と点や線のみであり，記述可能な可視化手法は限られている．また，GLO-STIX はグラフ構造の可視化に特化しているため，他のデータ構造に対する可視化手法の記述は困難である．

Lyra と iVis Designer には，可視化手法の構築部品が豊富に用意されており，利用者は，様々な可視化手法を簡潔に記述することができる．その点から，目標 1 を達成していると言える．さらに，これらのツールは，ドローツール風インタフェースを備えており，可視化手法の記述に必要な軸の位置や，マークの色等の視覚的なパラメータを直感的に記述することができる．利用者による編集の結果は可視化結果に即座に反映されるため，プログラミング言語と比べて分かりやすく，直感的と言える．そのため，目標 3，目標 4 について，プログラミング言語と比べて有効なユーザインタフェースである．しかし，これらのツールのユーザインタフェースには次のような問題点がある．まず，Visual Mappings 処理における，データとマークの対応関係を一覧することができないため，記述中の可視化手法の状態把握が困難である．2.3 節にて述べたとおり，実際の可視化手法の設計では，手法の記述の修正が繰り返し行われる．このことを考慮すると，データとマークの対応関係を一覧することができないことはユーザインタフェースの使いづらさにつながると考えられる．また，マークにデータを割り当てる際には，利用者は，マークを選択してパネルの表示内容を切り替えることが必要になることがある．このようなモーダルなユーザインタフェースは，利用者の本来の作業の妨げとなる [椎尾 10]．このような理由から，目標 3 について改善の余地があると考えられる．目標 2 に関しては，ツールのソースコードが公開されているものの拡張性を有しているとは言い難い．また，目標 5 に関しても，記述した手法をツール上や，ツールの提供するランタイムライブラリ上で実行することが前提となっており不十分と言える．

表 3.2 に既存のビジュアルツール，特に Lyra と iVis Designer における目標の達成状況を示す．

3.3 データフロービジュアル言語による記述

情報可視化手法は，データを視覚的な表現へと変換するためのデータの処理フローによって記述できる．例えば，散布図は，2 つの値を縦横の座標値へ変換し，その座標に円を描くという処理フローとして記述できる．データフロービジュアル言語は，データの処理フローを表す図（データフロー図）を描くことで，処理フローの記述を速く直感的に行うことができる記述方法である．すなわち，データフロービジュアル言語を用いたユーザインタフェースは目標 3 について有効な方法と考えられる．

汎用的なデータフロービジュアル言語を用いたシステムとして，Simulink[Sim]，LabVIEW[Lab]，Quartz Composer[Qua] が挙げられる．これらのシステムは，プログラミング言語相当の演算処理を記述することができ，様々なアプリケーションを制作できる．

ある分野に特化したデータフロービジュアル言語は，処理フローを記述する上で有効な手段であることが知られており [JHM04]，特化型の言語も開発されている．Haeberli ら [Hae88] はグラフィックアプリケーション構築のための，Bencina[Ben98] はオーディオプロセッシングシステム構築のための，Kobayashi ら [KST09] は Vjing のためのデータフロービジュアル言語を開発した．GraphEdit[gra] は，データフロー図を描くことで DirectShow のフィルタグラフをテストすることのできるツールである．Hansaki ら [HSMT06] は，データフロー図を描くことで SQL クエリを生成するツールを開発した．Zgraggen ら [ZZD14] は，棒グラフ等のチャートを連携させていくことで，データ分析を行うことのできるツールを開発した．Samuel ら [GGL+14] は，棒グラフや行列表現などの部品を組み合わせることで，チャートを構築しながらデータの分析を行えるツールを開発した．

科学的可視化の分野では，古くからデータフロービジュアル言語が用いられている．例えば，DataVis[Hil91]，AVS[UFK+89]，GeoVISTA Studio[TG02]，SRIRun[PJ95] が挙げられる．この分野では，大規模なデータを高速に処理することが求められている．これらのシステムは，非プログラマがデータフロー図を描くことで膨大なデータを高速に処理できるようにしている．

情報可視化の分野においても，データフロービジュアル言語は用いられている．North ら [NCS02] は，利用者がデータフロービジュアル言語を用いてマルチプルビューの可視化手法を制作できるシステムを開発した．GeoVISTA Studio は，科学的可視化向けのシステムではあるが，散布図や PCP[Ins85] などの手法を用いた可視化アプリケーションの構築もサポートする．

以上のように，様々なデータフロービジュアル言語が開発されているものの，多様な情報可視化手法の記述に着目したものは開発されていない．科学的可視化が対象とするデータは，物理的な位置や形状などの空間的性質を備えることが多いのに対して，情報可視化は空間的な性質を前提としない抽象的なデータを扱うため，表現上の自由度が高い．科学的可視化向けの既存システムでは，空間的性質を直接的にプロットするための座標系や，点や矢印などの構築部品が用意されているものの，それらの構築部品だけでは，例えば，配列系や連結系の記述が困難であり，情報可視化における表現上の自由度の高さに対応できない．そのため，科学的可視化向けのシステムは目標 1 に対して不十分である．North らのシステムや GeoVISTA Studio

は，棒グラフや散布図，PCP などの可視化ビューがあらかじめ用意されているものの，それらの可視化ビュー内の手法のカスタマイズは行えない．そのため，目標 1 に対して不十分である．目標 2 については，多くのシステムが部品の拡張機能を有している．例えば，GeoVISTA Studio では Java のクラスを実装することでその機能を拡張することができる．目標 5 については，AVS では，多くのプログラムに埋め込みが可能なようにクロスプラットフォームのランタイムライブラリが提供されている．GeoVISTA Studio には，Java のコンポーネントを出力する機能が備えられている．このように，プログラムへの埋め込みが考慮されているものの，目標 5 として設定したような描画 API や GUI ツールキットに依存しないというレベルには達していない．

表 3.3 に既存のデータフロービジュアル言語を用いたツールにおける目標の達成状況を示す．

表 3.1: プログラミング言語における目標の達成状況

目標 1	目標 2	目標 3	目標 4	目標 5
✓	✓			

表 3.2: 既存のビジュアルツールにおける目標の達成状況

目標 1	目標 2	目標 3	目標 4	目標 5
✓			✓	

表 3.3: 既存のデータフロービジュアル言語を用いたツールにおける目標の達成状況

目標 1	目標 2	目標 3	目標 4	目標 5
	✓	✓	✓	

3.4 時間軸を用いた量的データの可視化手法に関連する研究

時間軸を用いた量的データの可視化手法，すなわち，量的データ提示のためのアニメーションの利用方法に関連する研究として，静的な視覚的表現による量的データの提示という観点と，情報可視化におけるアニメーションの利用方法という観点から関連する研究を調査した．

3.4.1 静的な視覚的表現による量的データの提示

量的データ提示のための静的な視覚的表現に関して，量的データの表現精度に関する研究がなされている．Bertin[Ber84] は，値を表現することのできる視覚的な属性として視覚変数という考え方を提唱し，マークの位置や大きさ，色が量的データを表現できると述べている．

Cleveland ら [CM84] は，長さに比べて位置が比較的高精度に量的データを表現できることを示した．Mackinlay[Mac86] は，長さと向きが量的データを比較的高精度に表現できると述べている．小林 [小林 14] は，連結図を用いた重み付きグラフの表現において，エッジの持つ量的データを，連結図の線の太さや明度といった複数の視覚的な属性を用いて表現し，属性間の精度を比較する実験を行った．Guo ら [GHL15] は，連結図の線による不確実さの表現に関する実験を行った．

このように，量を表現することのできる静的な視覚的表現については研究がなされているものの，量的データ提示のためのアニメーションの利用方法やその精度については，未だ研究されていない．

3.4.2 情報可視化におけるアニメーションの利用

情報可視化において，アニメーションはよく用いられる手法の一つである．特に，Scatter-Dice[EDF08] や DiffAni[RJ13] のように，ビューの遷移にアニメーションがよく用いられている．また，アニメーションはカテゴリデータの提示には効果的であることが知られており，Munzner[Mun14] は，回転の向きや速さによってカテゴリデータを提示することができると述べている．

アニメーションの特性に関する研究もなされている．Archambault ら [APP11] は，スモールマルティプル表現 [Tuf90] とアニメーションの効果の比較を行った．Heer ら [HR07] は，棒グラフ，円グラフ，散布図などの一般的なチャートの間の遷移アニメーションの効果に関する研究を行った．Huber ら [HG05] は，背景との瞬きの頻度の違いによって，存在/非存在を表現できることを示した．

以上のように，情報可視化のためのアニメーションの利用方法に関する研究はなされているものの，量的データを提示するためのアニメーションの利用方法については研究がなされていない．

第4章　情報可視化手法記述のための データフロービジュアル言語を用いた ユーザインタフェース

本章では，情報可視化手法記述のためのデータフロービジュアル言語を用いたユーザインタフェースについて述べる．まず，データフロービジュアル言語の採用理由について説明する．次に，データフロービジュアル言語の仕様と，用意した構築部品について説明し，Iv Studio のユーザインタフェースについて説明する．構築した開発環境の評価として，従来形式のユーザインタフェースとの比較実験について述べたのち，データフロー図の描き方に関する調査について述べる．また，開発したデータフロービジュアル言語による可視化手法の記述例を紹介しながら多様性と簡潔性について述べ，最後に，Iv Studio の利用事例を示す．

4.1　データフロービジュアル言語の採用

データ，視覚値への変換方法，マークといった可視化手法の構築部品を組み合わせるビジュアルインタフェースは，多様な可視化手法の記述を可能としながらも，プログラミング言語よりも直感的な記述を可能とする記述方式である．構築する開発環境では，可視化手法の記述を容易にするために，そのようなビジュアルインタフェースを採用する．しかし，既存のビジュアルツールにはユーザインタフェースの使いやすさや，機能の拡張性について改善の余地がある．ユーザインタフェースの使いづらさの要因として，筆者は，既存のユーザインタフェースがデータの流れを開発者に提示していないという点に着目した．そして，開発者に対して，データの流れを視覚的に提示し，開発者がそのデータの流れを直接的に操作できるようにすることで，使いやすさを向上できると考えた．そこで，本研究では，データ，視覚値への変換方法，マークといった可視化手法の構築部品を，データフロービジュアル言語を用いて組み合わせる方式をユーザインタフェースに採用した（目標 3）[伊藤 15, 伊藤 16]．また，データフロービジュアル言語を導入することで，汎用的な演算の記述も可能となるため，拡張性についても改善が期待できる（目標 2）．一方，既存のデータフロービジュアル言語では，多様な可視化手法を簡潔に記述することができなかったが，可視化手法の構築部品を十分に用意することで，この点の改善も可能である（目標 1）．なお，目標 4 については，既存のビジュアルツールと同様に，ドローツール風のインタフェースを併用することで達成可能である．

表 4.1: 既存システムとの比較

システム	目標 1	目標 2	目標 3	目標 4
既存のビジュアルツール	✓			✓
既存のデータフロービジュアル言語を採用したシステム		✓	✓	✓
Iv Studio	✓	✓	✓	✓

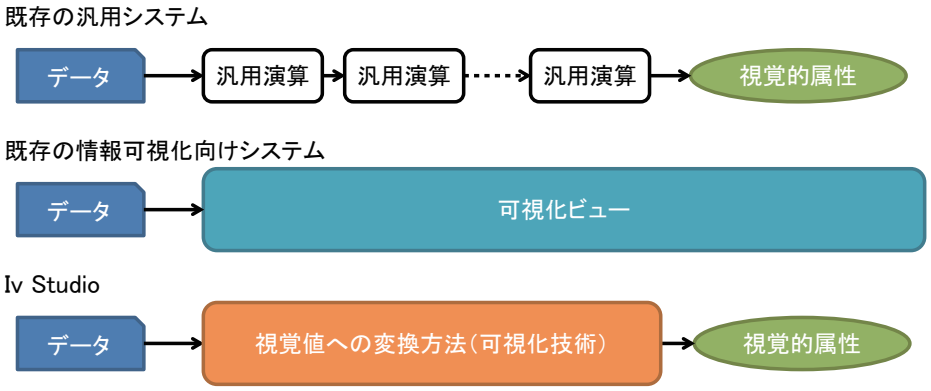


図 4.1: 既存のデータフロービジュアル言語を採用したシステムとの違い

4.2 情報可視化手法記述のためのデータフロービジュアル言語

3 章で述べたように，すでに，情報可視化手法の構築部品を組み合わせるシステムはいくつか提案されている．Iv Studio のユーザインタフェースにおける新規性は，そのようなシステムのユーザインタフェースにデータフロービジュアル言語を採用したことにある．その一方で，科学的可視化を対象にしたシステムや予め用意された可視化ビューを組み合わせるシステムに着目すると，すでにデータフロービジュアル言語を採用したものも存在するが，それらに比べると，可視化手法の構築部品を用意している点が新しい．目標 1 から目標 4 の達成という観点で，これら 2 種類のシステムと Iv Studio を比較した表を表 4.1 に示す．

既存のデータフロービジュアル言語を採用したシステムとの違いについて詳しく述べる．図 4.1 は，既存の汎用システムや，情報可視化向けシステムとの部品の用意の仕方のコンセプトの違いを説明した図である．Iv Studio は，視覚値への変換方法と，視覚的属性を持ったマークが部品として用意されていることで，汎用的なシステムよりも少ない記述量で，既存の情報可視化向けのシステムよりも多様な手法を記述できる点が異なる（目標 1）．既存の科学的可視化向けのシステムと比べた場合，Iv Studio には，カテゴリデータやグラフ構造，木構造を取り扱うための構築部品が用意されている点が異なる．

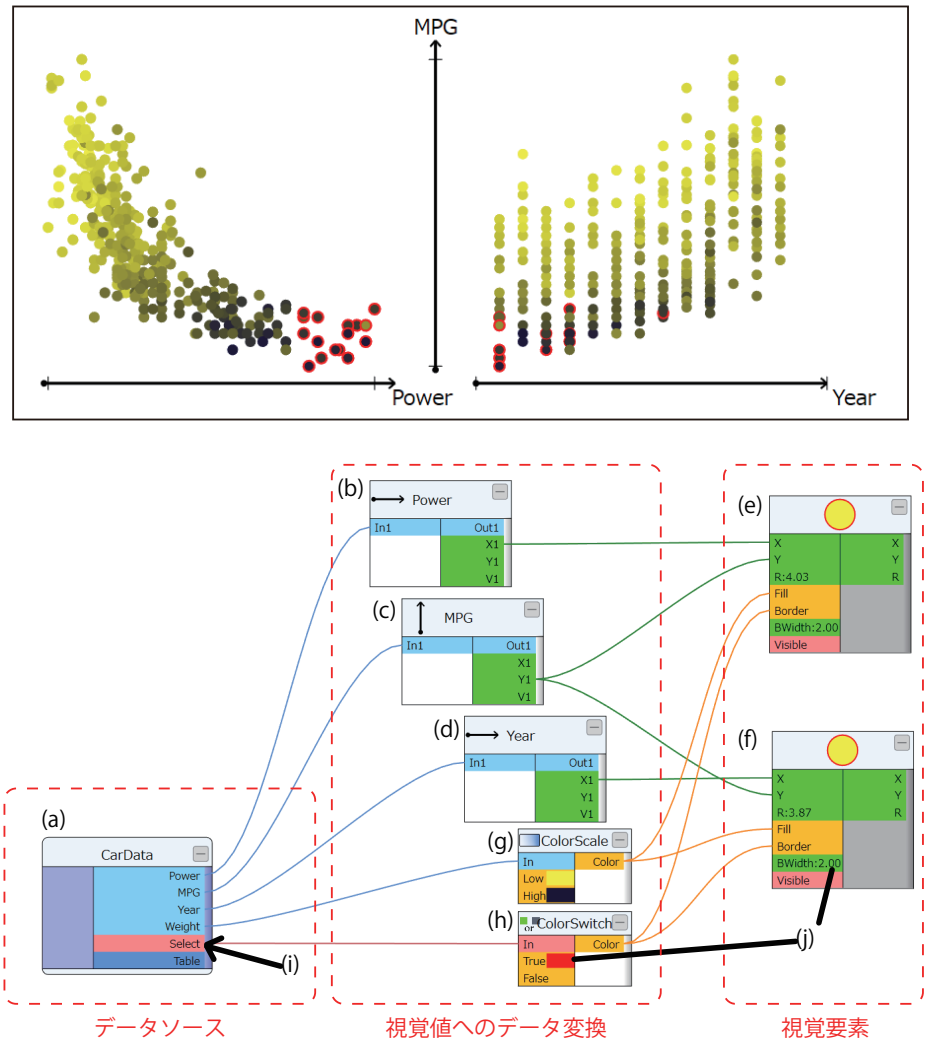


図 4.2: 開発したデータフロービジュアル言語による記述例

4.2.1 開発した言語による記述例

開発したデータフロービジュアル言語では，現在普及している情報可視化向けツールキットの D3[BOH11] と同様に，データドリブンのパラダイムを採用した．開発者は，データのレコード数に応じてマークを生成し，そのレコードの持つ値によるマークの視覚的属性の制御を，データフロービジュアル言語を用いて記述できる．

図 4.2 に，開発したデータフロービジュアル言語による記述例を示す．この図は，自動車の MPG（燃費），Power（馬力），Year（年式），Weight（車重）の 4 変量のデータ [car] を，上のように可視化する手法を記述した例であり，396 車種のデータを表している．この例で記述された手法は，縦軸に MPG を取り，その左に横軸が Power，右に横軸が Year の散布図を描画する．また，Weight を黄から黒のグラデーションを用いて表現する．さらに，レコードの

選択状態を円の境界線を赤くハイライトすることで表現する．

図 4.2(a) から (h) の矩形は，開発した言語の構築部品を表し，これらをノードと呼ぶ．ノードは，表 4.2 に示すようなノードの雛形（4.3 にて詳細に述べる）をもとに作られ，データの入力を受ける入力ピンと，処理したデータを出力する出力ピンを持つ．図 4.2(a) はデータソースを表すノード，図 4.2(b) から (d) と (g)(h) は視覚値へのデータ変換を表すノード，図 4.2(e)(f) はマークを表すノードである．開発者は，データソースを視覚値へと変換し，その値をマークによって表現する処理フローを表すデータフロー図を描くことで，可視化手法を記述することができる．

図 4.2(a) は，各出力ピンから，ピン名に対応するカラムのデータを出力する．図 4.2(b) から (d) は，単軸のノードである．単軸のノードは，入力ピン In1 に入力されたデータを軸上にマッピングし，X 座標を出力ピン X1 から，Y 座標を出力ピン Y1 から出力する．また，入力されたデータをそのまま出力ピン Out1 から出力する．図 4.2(e)(f) は，円を描くノードである．入力ピン X，Y に入力された座標に円を描く．入力ピン R は半径，入力ピン Fill は塗り色，入力ピン Border は境界線の色，入力ピン BWidth は境界線の太さを受け取るピンである．入力ピン Visible は，円の表示/非表示を切り替えるピンである．円の出力ピン X，Y，R は，描画する X 座標，Y 座標，半径を出力する．図 4.2(g) は数値を黄から黒の間の色に変換するノード，図 4.2(h) は真偽値を色に変換するノードである．

曲線はピンの接続関係を表す．図 4.2 では，縦軸 (c) に MPG，左の横軸 (b) に Power，右の横軸 (d) に Year を入力し，その出力座標を (e)(f) に入力することで，Y 軸を共有した 2 つの散布図を記述している．Weight を (g) に入力し，その出力を (e)(f) の Fill ピンに入力することで，Weight を黄から黒のグラデーションを用いて表している．また，(a) の Select ピン（レコードの選択状態の真偽値を出力する．4.2.3 にて詳細に述べる.）を (h) に接続し，その出力を (e)(f) の Border ピンに入力することで境界線のハイライトを記述している．(h) は，入力が真のときに赤，偽のときに透明を出力するように設定されている．

4.2.2 開発したデータフロービジュアル言語の仕様

ノードは一つ以上の関数からなり，さらにそれらの挙動を変更するためのパラメータを持つことがある．例えば，図 4.2(b) から (d) のノードはそれぞれ単軸を表し，それらの位置や向き，長さなどはパラメータによって決められる．また，図 4.2(h) の出力する色もパラメータによって決められる．

関数は，複数の入力ピンと出力ピンを持ち，入力ピンから入力された値を処理し，出力ピンから出力する．出力ピンは複数の入力ピンと接続できるが，入力ピンは 2 つ以上の出力ピンと接続できない．

プログラミング言語で考えた場合，ノードの雛形はクラス，ノードはクラスのインスタンス，パラメータはクラス変数，関数部分はメソッド，入力ピンは引数，出力ピンは戻り値に相当する．ピンにはデータ型が設定されており，異なるデータ型のピンを接続することはできない．プログラミング言語によるノードの雛形の拡張を容易にするために，一般的な手続

き型のプログラミング言語に対応するような設計としている（目標 2）．プログラミング言語を用いたノードの拡張については，5 章にて詳細に述べる．

ノードは矩形で表現され，上部のヘッダ部分と，その下の関数部分から構成される．接続関係を表す曲線の色は，接続しているピンのデータ型を表す．

ノードのヘッダ部分には，ノードの持つ関数の挙動を連想できるよう，ノードの接続状態やパラメータを反映させた図を表示する．例えば，単軸のノード（図 4.2(b) から (d)）は，単軸のアイコンを表示し，その横に割り当てられているデータの情報を表示する．マークのノードは，そのノードの描くマークを表示する（図 4.2(e)(f)）．

関数部分には，左側に入力ピンを，右側に出力ピンを表示する．データの種類を読み取りやすくし，接続できないピンを分かりやすくするため，ピンの色によってデータ型を表す．入力ピンには，接続されていない場合のデフォルト値が指定されており，未接続のピンはその値を表示する（図 4.2(j)）．

4.2.3 インタラクション手法の記述

インタラクション手法の記述のためには，可視化結果に対する閲覧者の入力を受け付け，それによって可視化結果を変化させるための仕組みが必要となる．そこで，可視化のためのデータの流れとは逆方向に，マークからデータソースにレコードの選択状態を伝搬する仕組みを導入した．開発者は，レコードの選択状態をデータソースのノードの Select ピン（図 4.2(i)）から取得することができる．Select ピンを利用することで，例えば，選択状態にある項目を別のビューでもハイライトする機能（Linking & Brushing[Kei02]）を記述できる．

4.3 ノードの雛形

既存の情報可視化向けのツールキット [HCL05, BH09] や，ビジュアルツール [SH14, RHY14] により，可視化手法の構築部品として，視覚値への変換方法とマークを用意することで，多様な可視化手法を簡潔に記述できることが示されてきた．データフロービジュアル言語においても，これらの先行研究にならうことで，多様な手法を簡潔に記述できるようになると考えられる．本節では，目標 1 で定めた範囲を基準に整理した，ノードの雛形の用意の仕方，すなわち，ノードの雛形のライブラリの設計について述べる．

表 4.2 に，用意した主なノードの雛形を示す．本節では，これ以降，雛型の種類ごとに設計の詳細を述べていく．

4.3.1 データソースとデータ処理

データソースに関するノードの雛形には，Visual Mappings 処理への入力である，データテーブルを用意した．この雛形により，多次元データの取り扱いを可能としている．データテーブルのノードは，データソースとなる表データのカラムに対応するピンと，4.2.3 にて述べた

表 4.2: 用意したノードの雛形

雛形の種類	ノードの雛形
データソース	データテーブル
データ処理	グルーピング，他テーブルの参照，集約，フィルタリング，ソート
マーク	矩形，円，扇，直線，折れ線，多角形，文字列
視覚値への変換	単軸，直交座標軸，極座標軸，スケールメモリ，カラーマッピング
演算処理	汎用演算（四則演算，比較演算子），色演算，幾何演算

Select ピン，テーブル情報を出力する Table ピンを持つ．Table ピンは，後に述べるグルーピングのノードで使用される．開発した言語では，一度に複数のデータテーブルのノードを作成することができる．そのため，開発者は，複数の表データを使った可視化手法の記述も可能である．グラフ構造や木構造については，グラフの頂点やエッジに関するデータテーブルを読み込むことで取り扱いを可能としている（詳細は，4.3.3 にて述べる）．

データ処理に関するノードの雛形には，まず，複数のデータテーブル間の関係性を記述できるようにするために，グルーピングや他テーブルの参照のための雛形を用意した．情報可視化では，あるカラムの値を基準としてグループ化したものを可視化することが行われる．このような手法の記述を可能とするために，グルーピングを行うノードの雛形を用意した．また，関係データベースのように，複数のデータテーブルがあるカラムの値に基づいて関係づけられている場合には，その関係の可視化も取り扱えるように，他のテーブルの値を参照することができる雛形を用意した．次に，集約処理（平均値などの統計値の算出）や，データのフィルタリング，ソートのためのノードの雛形も用意した．開発した言語は，Visual Mappings 処理の記述を主としているため，事前に可視化しやすい形に加工されたデータテーブルが入力されることを想定している．そのため，これらの雛型は Visual Mappings の記述においては，本来不要のものではあるが，手法の設計プロセスにおける試行錯誤に便利なため用意した．

4.3.2 マーク

マークのノードの雛形には，SVG の要素や一般的な描画 API の持つメソッドを参考にし，矩形，円，扇，折れ線，文字列，直線，多角形といった図形の雛型を用意した．直線によって連結系の記述を，多角形によって領域系の記述を可能としている．

既存のビジュアルツールでは，このような図形の他に，例えば，折れ線グラフを記述するための折れ線のような，特定の用途を想定した変換処理も含むようなマークの構築部品が用意されている．この理由は，複雑な処理フローの記述が困難なためと考えられる．一方，Iv Studio では，データフロービジュアル言語により複雑な処理の記述も比較的容易なため，マークのノードの雛形としては「図形を描く」機能のみを持たせたシンプルなもののみを用意し，開発者が，マークのノードを選ぶ際に，描きたい図形のことのみを考えれば良いようにした．

マークのノードの雛形では，それぞれの図形の定義に必要な座標や大きさの入力ピン以外

に，可能な限り以下の入力ピンを設けた．

- 塗り色（Fill）
- 枠線の色（Border）
- 枠線の太さ（BWidth）
- 表示／非表示の切り替え（Visible）

全てのマークが色を使った値の表現ができるように，その色を制御するための入力ピンを必ず設けた．塗りつぶしと枠線のマークを分けることも検討したが，記述にかかるノード数を減らすために，1 つのマークのノードで塗りつぶしと枠線の両方を担うようにした．また，図形の表示／非表示を切り替えるための入力ピン Visible を設けた．開発者は，Visible ピンとデータソースの Select ピンを組み合わせることで，選択したレコードのみラベルを表示するようなインタラクティブな手法を記述できる．

マークの出力ピンには，複合マーカーを簡潔に記述できるように，描画位置等を出力する出力ピンを設けた．ここで言う複合マーカーとは，1 レコードのデータを提示するための複数のマークで構成された視覚的な要素のことを指す．例えば，円とその周りを囲む環状のマークから構成される視覚的な要素や，ラベル付きのマークなどを指す．開発者は，マークに設けられた出力ピンを利用することで，マークの相対位置に別のマークを描くような手法も容易に記述できる．

4.3.3 視覚値への変換方法

まず，多次元データ可視化のために用意した視覚値への変換方法のノードの雛形について説明する．表現の多様性を確保するため，データテーブルのピンに用いられるデータ型と，マークの持つ視覚的属性を基準として，変換方法の雛型を用意した．

データテーブルのピンに用いられるデータ型は主に，数値型，文字列型，真偽値型の 3 種類である．その他にも，データテーブルは，時間型のデータを持つこともできるが，時間型については，数値型に変換する雛形を用意することで数値として取り扱うことができるようにしているため，考慮の対象から除外した．文字列型と真偽値型はともにカテゴリデータを表すデータ型であるが，文字列型は視覚値へ変換する際に，2 種類以上の値を想定しなければならないのに対し，真偽値型は真と偽の 2 種類のみを想定すればよいという違いがある．そのため，この 2 つの型は別のものとして取り扱った．

マークの持つ視覚的属性は，位置，大きさ（半径，幅，高さ），色，角度の 4 種類である．これらの視覚的属性 4 種類に対して，データソースのデータ型（数値型，文字列型，真偽値型）の値を変換できるようにするために，視覚値への変換方法の雛形を網羅的に用意した（表 4.3）．

位置への変換には，全て軸のノードを利用するようにした．ただし，量的データを受け取る場合と，カテゴリデータを受け取る場合とで，データのマッピング方法が異なる．開発者

表 4.3: 用意したノードの雛形

	位置	大きさ	色	角度
数値型	軸	スケールメモリ	カラースケール	極座標軸
文字列型	軸	ー	カラーマッパー	極座標軸
真偽値型	軸	サイズスイッチ	カラースイッチ	極座標軸

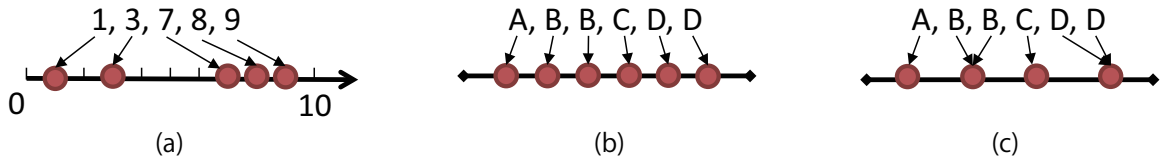


図 4.3: 軸のデータマッピング方法

は，量的データを受け取る場合には，軸上にその数値を射影するようにデータをマッピングする方法（モード A : 図 4.3(a)）を利用することができる．一方，カテゴリデータを受け取る場合には，レコード順に均等に並べるようにマッピングする方法（モード B : 図 4.3(b)）や，値の内容ごとに均等に並べるようにマッピングする方法（モード C : 図 4.3(c)）を利用できる．モード A を利用することで，座標系の表現を記述でき，モード B やモード C を用いることで，配列系の表現を記述できる．軸については，可視化手法で用いる頻度の高い，直交座標軸の雛形も用意した．直交座標軸においても，X 軸，Y 軸それぞれについてマッピング方法を変更できる．

大きさへの変換には，数値型ではスケールメモリを，真偽値型ではサイズスイッチを利用するようにした．文字列型から大きさへの変換については，閲覧者に誤解を与えるため推奨されない表現 [Mac86] であることから用意しなかった．スケールメモリは，大きさの凡例を描き，その範囲に数値をマッピングして，大きさを出力する．サイズスイッチは，真偽値を受け取り，その真偽値に応じて，ノードのパラメータに指定された大きさを出力する．開発者は，サイズスイッチを利用することで，選択したレコードに対応する図形の大きさや線の太さを変更するといった手法を簡潔に記述できる．

色への変換に関しては，データ型によって異なるノードの雛形を利用するようにした．数値型については，ノードのパラメータに指定された 2 つの色を端点として，その間のグラデーションで数値を表す色を出力するカラースケールを用意した．文字列型については，文字列を色相の違いで表すような色を出力するカラーマッパーを，真偽値型については，ノードのパラメータに指定された 2 つの色を，入力された真偽値によって切り替えて出力するカラースイッチを用意した．情報可視化において，色は，文献 [Ber84, Mun14] で述べられているように，量的データは明るさや彩度，カテゴリデータは色相を用いることが推奨されている．用意した雛型は，これらの文献に基づいているが，カラースケールやカラースイッチについては，開発者が自由な配色を行えるようにするため，基準となる色の指定に制約を設けなかった．

角度への変換には，全て極座標軸のノードを利用するようにした．極座標軸のノードも，軸

のノードと同様のマッピング方法を用意した．

次に，グラフ構造を取り扱うためのノードの雛形について述べる．開発した言語では，グラフの頂点のデータテーブルとエッジのデータテーブルの 2 つのデータテーブルと，その間の関係を指定するのできるノードの雛型によって，グラフ構造を取り扱えるようにした．このグラフ構造の取り扱い方は，リレーショナルデータベース等で一般的な方法である．開発者は，4.3.1 で述べた，他テーブルの参照のノードを利用することで，2 テーブル間の参照関係を記述することができる．そのほか，頂点のデータテーブルと，エッジのデータテーブルから参照関係を受け取り，力指向のアルゴリズム [Ead84] によってグラフのノード位置を求め，求めた位置を出力するノードの雛形が用意されており，開発者は，この雛型を用いて，網図形式のグラフの可視化手法を記述できる．

最後に，木構造を取り扱うためのノードの雛形について述べる．データテーブルの 1 レコードを木の 1 頂点とみなし，頂点のユニークなキーを保存するカラムと，親の頂点のキーを保存するカラムを持ったデータテーブルを木構造として取り扱えるようにした．木構造のレイアウト技術として，頂点と頂点の親のキーを受け取り，樹状図の形にレイアウトするノードを用意した．

グラフ構造や木構造の取り扱い方法をデータテーブルに基づいた方法にしたことで，頂点やエッジが重みを持っていた場合に，開発者がその重みを多次元データの表現のために用意したノードの雛形を用いて表現できるようにした．

4.3.4 演算処理

目標 2 への対応のひとつとして，汎用演算，色演算，幾何演算等の演算処理のためのノードの雛形も用意した．これらの雛形により，独自のレイアウト技術やカラーマッピング技術，マークの形状の記述も可能としている．

開発当初は，四則演算と算術関数，色の生成の雛形のみを用意していた．しかし，これらの雛形のみで，可視化技術を記述しようとすると，記述が複雑になることが分かった．そこで，ベクトル演算や凸包算出，重心の計算などの幾何演算と，色の混色や明るさなどの一部のパラメータをコントロールする色演算のための雛形も用意した．

4.4 Iv Studio のユーザインタフェース

本節では，Iv Studio のユーザインタフェースを説明する．まず，画面構成と基本的な操作方法について述べ，データフロー図の記述を容易にするための支援機能を紹介する．

4.4.1 ユーザインタフェースの概要

Iv Studio の画面構成を図 4.4 に示す．ノードメニュー（図 4.4(a)）には，ノードの雛形が上部のタブにまとめられている．開発者は，ノードメニューからノードの雛形を選択し，編集

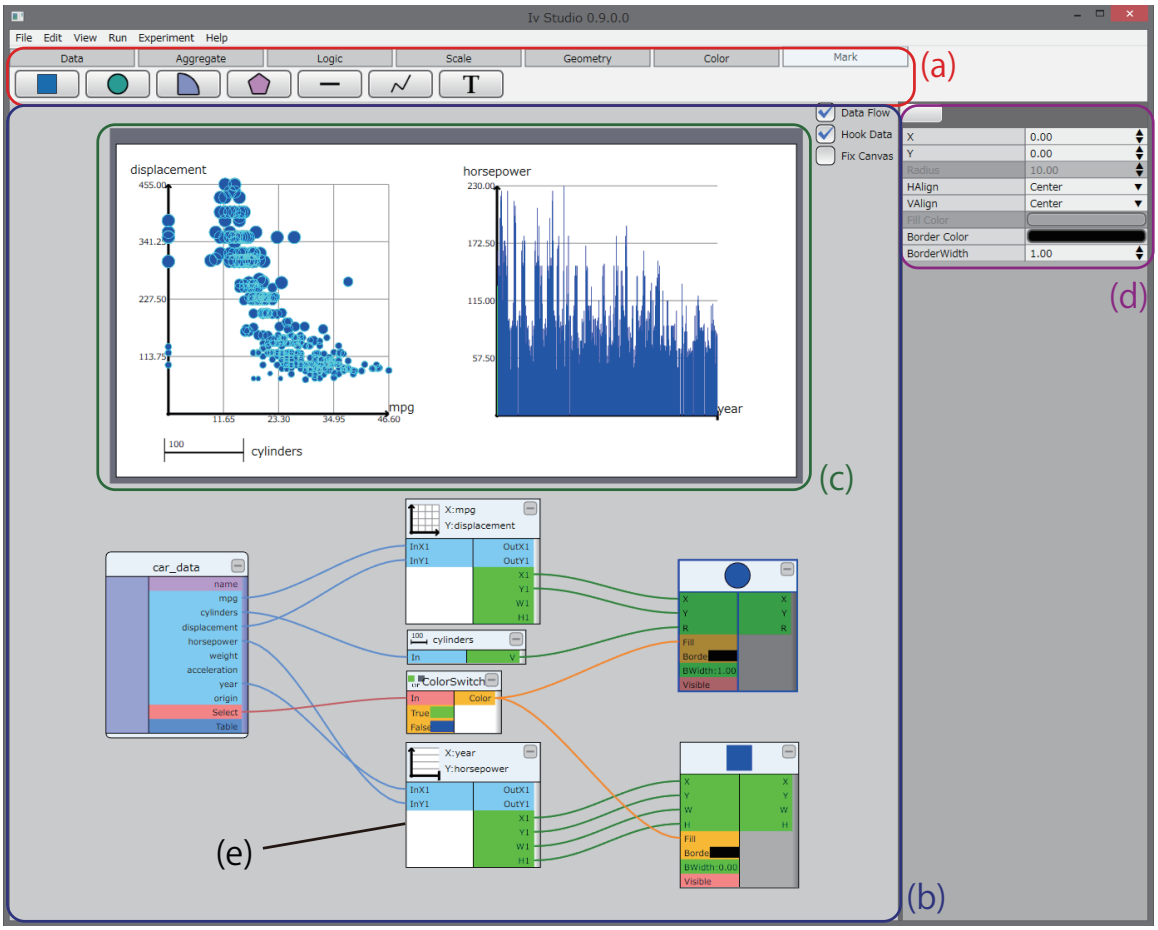


図 4.4: Iv Studio の画面構成

領域（図 4.4(b)）をクリックすることでノードを作成できる．ドラッグ操作によってピンを接続することができ，ノードのヘッダ部分をドラッグすることでノードを移動できる．ノード内の未接続の入力ピンをクリックすることで，デフォルト値を編集できる．また，パラメータパネル（図 4.4(d)）から，ノードのパラメータを編集できる．キャンバス（図 4.4(c)）には，記述中の可視化手法のプレビューが表示され，編集結果は即座に反映される．

Iv Studio のユーザインタフェースでは，先行研究の Lyra や iVis Designer の良い点を積極的に継承した．開発者は，両方の先行研究と同様に，ドローツール風インタフェースにより視覚的なパラメータを直感的に編集できる（目標 4）．例えば，キャンバス上の図形に表示されるハンドルを使うことにより，位置や大きさを直感的に調整できる（図 4.5）．また，iVis Designer のように，ノード生成時にキャンバス上のマークをクリックすることで，ピンが接続された状態でノードを作成できる．さらに，Lyra のように，マークに対して出力ピンをドラッグアンドドロップすることで，ピンを接続することもできる．

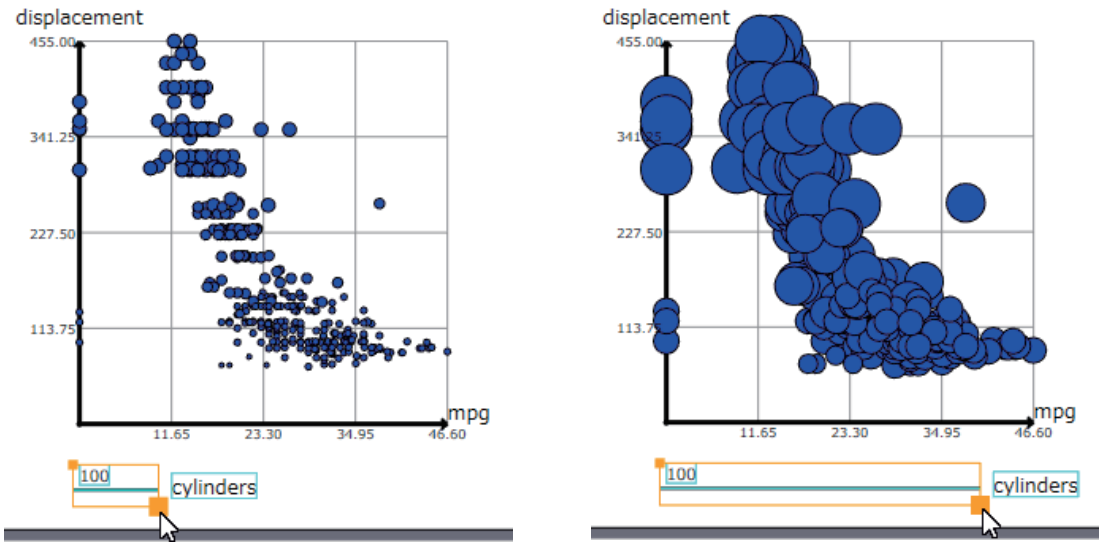


図 4.5: ハンドルによる大きさの変更

4.4.2 データフロー図の記述支援機能

開発した言語は，可視化手法の構築部品を組み合わせる方法を採用しているため，可視化ビューをノードとして用意する場合と比べて，記述に要する操作量が多い．そこで，その操作量を減らすために，2 つの支援機能を用意した．

一つ目は，開発者が異なるデータ型のピンを接続しようとした際に，型変換を行うためのノードを自動的に挿入する機能「型キャスト」である．例えば，散布図の円の色に数値を割り当てたい場合，データソースのノードの出力ピンを円のノードの入力ピン Fill につなぐことで，数値を色へ変換するノードが自動挿入され，割り当てが完了する．このようなノードの自動挿入機能は，GraphEdit[gra] にも搭載されている機能である．ただし，情報可視化では，1 つの可視化手法の中に，同じデータから同種類の視覚的属性へ変換方法が複数あることは好ましくない [SH14] ため，すでに変換用のノードが作成されている場合は，そのノードからの出力をつなぐようにした．この点が，GraphEdit の自動挿入機能とは異なる．この機能は，単に記述に要する操作量を減らすだけではなく，開発者の「円の色でデータを表したい」という意図をより直接的に入力することができるようにするという効果も期待できる．

もう一つは，関数間のピンをまとめて接続する機能「関数接続」である．開発者は，ノードの出力ピンの右側のスペースをドラッグし，入力側の関数部分にドロップすることで，X と X1 や Y と Y1 のように名前の類似するピンをまとめて接続できる．

4.5 ユーザインタフェースの評価実験

本研究における技術的な課題は，ユーザインタフェースにおける手法の記述のしやすさ（目標 3）の向上である．そのためのアプローチとして，データフロービジュアル言語をユーザインタフェースに採用した．そこで，データフロービジュアル言語の採用が記述のしやすさを向上させたかどうかを確かめるため，データフロービジュアル言語を用いたユーザインタフェース（以下,「DI」と呼ぶ）と，既存のビジュアルツール形式のユーザインタフェースとを比較する被験者実験を行った．この実験では，以下の 4 点を調査した．

調査項目 1 DI は理解しやすいか？

調査項目 2 DI は記述を速く行えるか？

調査項目 3 DI はカスタマイズの記述に好まれるか？

調査項目 4 DI はカスタマイズの記述を速く行えるか？

これらの項目によって，目標 3 に対する評価を行った．項目 2，4 は，効率的に記述できるかどうかに関する調査項目であり，項目 1，3 は，記述方法を直感的に理解できるかどうかに関する調査項目である．カスタマイズの記述の場合には，カスタマイズのための特別な学習は必要ではないものの，修正箇所の把握など，はじめから手法を記述する場合とは異なる作業が必要となる．そのため，項目 3 については，直感的に理解しやすいものが好まれるであろうという推測に基いて，DI が好まれるかどうかを調査した．

なお，目標 4 については，ドローツール風インタフェースを採用することで対応を行ったが，これについては先行研究によってその有効性が示されていることから，本研究では改めて評価は行わなかった．また，記述パラダイムには，既存のビジュアルツールと同様のパラダイムが採用されているため，可視化手法の考案が容易か否かといったパラダイムの容易さについても改めて評価しなかった．

4.5.1 比較対象のユーザインタフェース

Iv Studio が既存のビジュアルツールと異なる点は，開発者に対して記述中の可視化手法のデータの流れを提示していることと，その流れを直接的に操作できることである．それらの効果を評価するため，DI とデータフロー図を提示しない記述インタフェースとを比較することにした．そのようなインタフェースとして，図 4.6 のようなブロックインタフェース（以下,「BI」と呼ぶ）を Iv Studio 上に実装した．BI は，先行研究の中でも直感的なユーザインタフェースを持つ Lyra を模して設計した．Lyra をそのまま利用せず，Iv Studio 上に BI を実装した理由は 2 つある．一つは，Lyra では Linking & Brushing の記述が行えないことや，Lyra と Iv Studio では視覚値への変換方法の用意の仕方が異なるといったように，機能面に若干の違いがあるためである．もう一つは，被験者にどちらのユーザインタフェースが提案手法なのかを分からないようにするためである．BI は，Lyra による可視化手法の制作操作を完全に

再現するものではないが，DI のデータフローを開発者に見せる効果とデータフロー図の編集というユーザインタフェースの効果を調べるという目的においては，比較対象となるユーザインタフェースである．

Lyra には，選択したマークの色や大きさを設定するためのパネルが備えられている．開発者は，そのパネルに，関連付けたいデータや視覚値への変換方法を表すブロックを，画面左のパネルからドラッグアンドドロップすることで可視化手法を記述できる．Lyra のブロックは，Iv Studio のノードに相当する．そこで，筆者は，Lyra のブロック配置の操作と同様の操作によってノードのピンを接続できるように，次のようなユーザインタフェースを実装した．

左パネル上部にデータのノードの出力ピンのリスト，すなわち，データカラムのリストを表示するデータパネル（図 4.6(a)），下部に座標軸などのデータ変換用のノードのリストを表示する変換ノードパネル（図 4.6(b)）を設置した．右パネル上部にマーク用のノードのリストを表示するマークノードパネル（図 4.6(c)），下部にピンの接続を行うためのブロックパネル（図 4.6(d)）を設置した．変換ノードパネルとマークノードパネルを合わせてノードパネルと呼ぶ．データパネルは，データカラムのカラム名を，ノードパネルはノード名とそのノードの種類を表すアイコンを表示する．軸のノード（図 4.6 の PlotField1 等）のようにデータ変換とマークの両方の機能を持つノードについては，(b) と (c) の両方に表示する．ブロックパネルには，選択中のノードの入力ピンを表示し，入力ピンの領域には，データがどのノードを経由して変換されてきたかを表示する．開発者は，ノードパネルの項目をクリックすることで，そのノードを選択できる．また，接続先のノードをノードパネルから選択してブロックパネルにそのノードの入力ピンを表示し，ブロックパネルの入力ピンの領域に，図 4.6 の矢印のようにデータパネルやノードパネルから項目をドラッグアンドドロップすることでピンを接続できる．出力ピンを指定する必要はなく，Lyra と同様にノードをドロップするだけでピンの接続を行える．

BI を用いた場合も，ドローツール風のインタフェースや，型キャスト，関数接続も利用可能である．ただし，ユーザインタフェースの違いの比較を行うために，ノードの接続操作やブロックパネルの操作を介さずにピンを接続できるような機能は，BI，DI ともに利用できないようにした．例えば，Iv Studio には，マークに対して出力ピンをドラッグアンドドロップすることでピンを接続する機能があるが，実験ではこの機能は利用できないようにした．

4.5.2 実験 1

はじめに，被験者に対して，情報可視化に関する専門用語の解説と可視化手法の記述の仕方，すなわち，ビジュアルエンコーディングの考え方の説明を行った．その説明には，データフロー図のような図を一切用いず，例となる視覚的表現と文章のみで説明を行った．

次に，表 4.4 のようなタスクを行ってもらった．カウンターバランスのため，半数の被験者には，DI の練習，1a，BI の練習，1b，2a，2b，3 の順でタスクを行ってもらい，残りの被験者には，BI の練習，1b，DI の練習，1a，2b，2a，3 の順で行ってもらった．タスク中は，Iv Studio で被験者の操作履歴を時刻とともに記録した．被験者には，Yes/No で答えられる質問には応じるが,「どのように作るか？」といった質問には応じないと事前に伝えた．また，タ

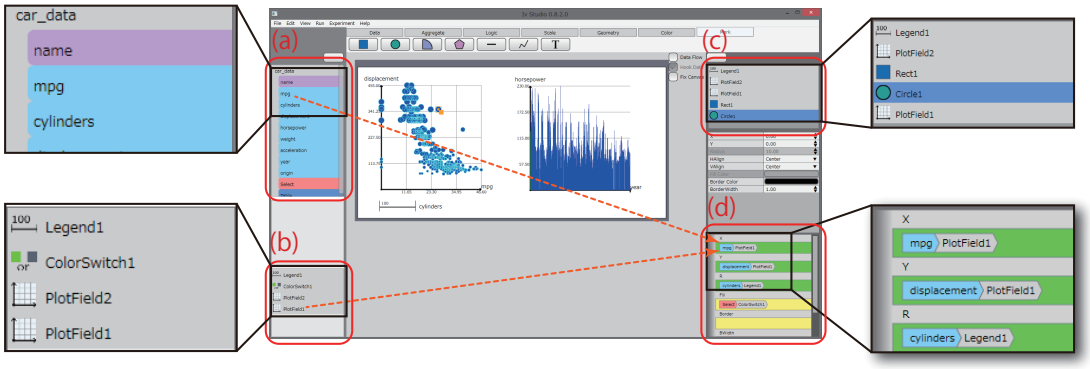


図 4.6: ブロックインタフェース

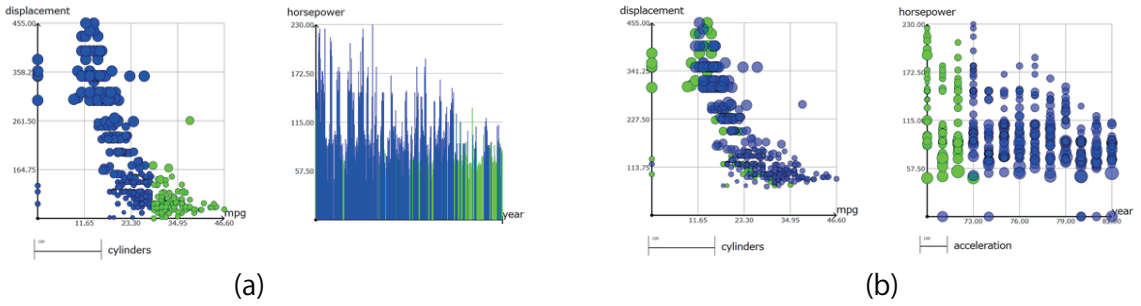


図 4.7: 実験 1 にてタスクの説明時に被験者に提示した図

スク 1 の練習で使用した資料を参照できるようにした．タスクは，単にバブルチャートや棒グラフを完成させることであり，図の配置や配色を厳密に同じにする必要はないと伝え，タスクをできるだけ速く達成するように求めた．そして，被験者がタスクを達成したと判断した時点で，その旨を被験者に宣言をしてもらった．実験者は，必要なノードが生成され，それらが適切に接続され，ノードのパラメータが適切に設定された状態をタスク達成とみなした（詳細な達成基準については，後述する）．タスク未達成にもかかわらず被験者が終了を宣言した場合は，実験者から未達成の旨を伝えるようにした．実験には，27 インチ（解像度 2560 × 1440）のディスプレイを用い，全被験者が同じ PC を使用した．全タスク完了後，主観的な評価を得るために，5 段階のリッカート尺度（-2 から 2）で回答するアンケートを行い，質問項目に対して特記事項があれば自由に記述してもらった．実験は，一人の被験者あたり，1 時間から 1 時間半程度で完了することを想定した．

タスク 1 の練習では，はじめに操作方法を解説するチュートリアルを再生し，その後，画面の指示通りに操作するトレーニングモードを行ってもらった．チュートリアルは，ユーザインタフェースの画面の説明や，操作方法を説明するものである．被験者には，チュートリアルを通して，基本的なユーザインタフェースの操作方法を学習してもらった．トレーニングモードは，目的の手法を制作する手順が 1 操作ごとに画面に表示され，その指示に従い操

表 4.4: 実験のタスク

タスク	ユーザインタフェース	タスクの内容
1a	DI	DI の練習後，図 4.7(a) の手法を制作
1b	BI	BI の練習後，図 4.7(a) の手法を制作
2a	DI	図 4.7(a) の手法を制作
2b	BI	図 4.7(a) の手法を制作
3	DI+BI	図 4.7 を (b) のようにカスタマイズ

作することで，被験者が手法の制作を体験することのできるものである．被験者には，このモードを通して，散布図，棒グラフ，値を円の大きさで表す表現手法，Linking & Brushing の記述を完了するまで体験してもらった．チュートリアルやトレーニングの内容は，紙に印刷し，被験者に渡した．

タスク 1，2 では，自動車のデータを図 4.7(a) のように可視化する手法を制作してもらった．左にバブルチャート，右に棒グラフを制作し，左右の図を Linking & Brushing を用いて連携してもらった．図 4.4 のデータフロー図は，この可視化手法を記述したものである．このデータフロー図のノードの接続状態を達成基準とした．ノードのパラメータについては，棒グラフを作るための直交座標軸のノード（図 4.4(e)）の X 軸のデータのマッピング方法が，均等に並べる方法に設定され，そのノードの Y 軸ともうひとつの直交座標軸のノードのマッピング方法が数値を射影する方法に設定されていることを達成基準とした．タスク 1，2 ともに，全被験者が DI，BI をそれぞれ 1 回ずつ用いて合計 2 回，同じ手法を制作するタスクを行ったが，用いるインタフェースの順番については，半数の被験者は DI を先に用い，残りの被験者は BI を先に用いるという違いがあった．タスク 1 では，どちらのインタフェースが理解しやすいかを比較するためために，2 回のタスクうち 1 回目のタスクの達成時間を比較した（調査項目 1）．一方，タスク 2 では，どちらのインタフェースが被験者の思い描いたビジュアルエンコーディングの方法を速く記述できるかを比較するため，被験者ごとに DI と BI の達成時間の比較を行った（調査項目 2）．つまり，タスク 2 では，エンコーディング方法の考案などの時間を含めない，純粋なユーザインタフェースの速さの比較を行った．

タスク 3 では，被験者の好みのユーザインタフェースを自由に使い，タスク 2 終了後の状態から図 4.7(b) のような手法にカスタマイズしてもらった．具体的には，右図をバブルチャートへ変更し，円の色を半透明に変更してもらった．図 4.8 にタスク 3 のノードの接続状態の達成基準となるデータフロー図を示す．ノードのパラメータについては，2 つの直交座標軸のノードのマッピング方法がすべて数値を射影する方法に設定され，図 4.8 中央のカラースイッチノードの False 時の色に半透明色が設定されていることを達成基準とした．被験者は，画面上のチェックボックスからデータフロー図の表示を切り替えることができる．タスク 3 は，データフロー図を非表示の状態から開始した．つまり，DI にとって不利な状態から開始しており，被験者が DI を利用するためには 1 つ操作を行う必要があった．タスク 3 では，カスタマイズするタスクにおいて，被験者が DI を好んで利用するかを調査した（調査項目 3）．さ

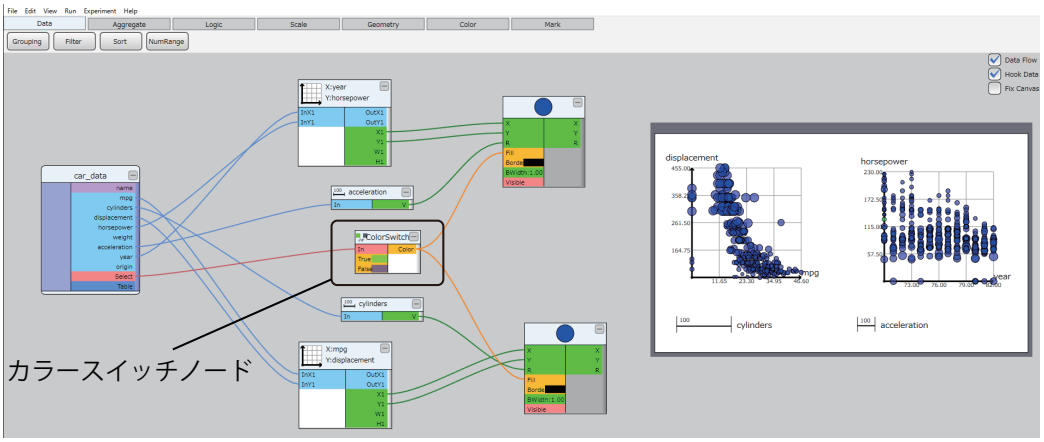


図 4.8: タスク 3 の達成基準となるデータフロー図

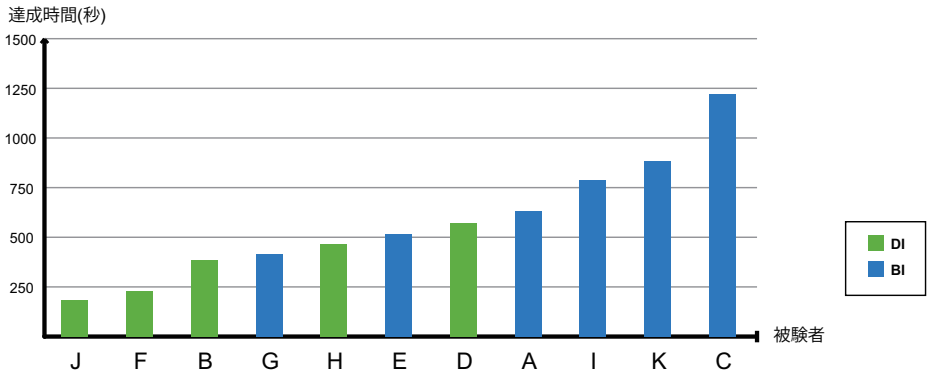


図 4.9: タスク 1 の被験者ごとの達成時間

らに，タスクの達成時間を記録し，DI を選んだ被験者と BI を選んだ被験者とで達成時間を比較することも予定した（調査項目 4）．

被験者は，情報科学を専攻する学生及び大学院生（10 名）とソフトウェア開発企業に務める技術者（1 名）の合計 11 名で，全員にプログラミング経験があった．また，大学院生のうち 1 名が，情報可視化を専門とする研究室に 1 年以上在籍している学生であった．5 名の被験者には DI を先に用いてもらい，残りの 6 名には BI を先に用いてもらった．

4.5.3 実験 1 の結果

図 4.9 に，タスク 1 で 2 回行ったタスクのうち 1 回目のタスクの達成時間を被験者ごとに示す．緑が DI，青が BI での達成時間を示す．タスク開始時刻から最後に行った操作の時刻までをタスクの達成時間とした．達成時間の中央値を求めたところ，DI が 382.772 秒，BI が 709.426 秒であった．サンプル数が少なく，分布の仮定が困難なことから，ウィルコクソンの

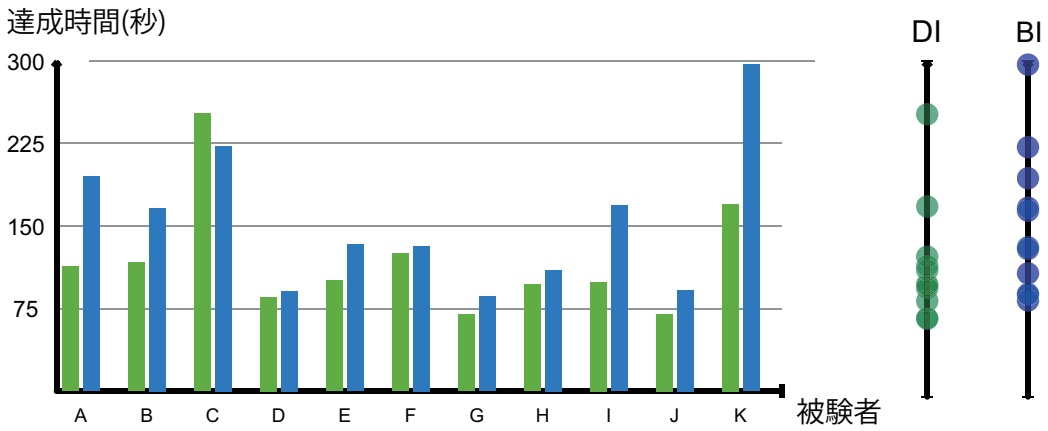


図 4.10: タスク 2 の被験者ごとの達成時間

順位和検定を行ったところ，有意水準 5%で有意差が認められた (T = 18)．タスク 1 の結果から，初めて利用する場合において，DI の方が可視化手法を速く記述できると言える．

図 4.10 に，被験者ごとのタスク 2 の達成時間を示す．右の図は，それぞれの達成時間を直線上にプロットした図である．タスク 2 は記述の仕方に慣れた状態での遂行を想定していたが，被験者 C はタスク 2 中でも資料の参照に 1 分以上の時間を要していた．このことから，被験者 C のデータは異常値とみなし，以後の分析では残り 10 名分のデータを対象とした．10 名の達成時間の中央値を求めたところ，DI が 99.219 秒，BI が 132.003 秒であった．この 10 名について，サンプル数が少なく，分布の仮定が困難なことから，ウィルコクソンの符号順位検定を行ったところ，有意水準 5%で有意差が認められた（T = 0）．くわえて，DI が BI に比べてどの程度速くなったかを計算した．その結果の中央値は 23.7%であった．タスク 2 の結果から，DI の方が可視化手法を速く記述できると言える．

タスク 3 では，すべての被験者が，データフロー図を表示し，以降 DI を用いてタスクを行っていた．タスク完了後に，なぜ DI を用いたかを被験者に尋ねたところ「構造が直感的に見える」や「データフローのほうが使いやすかった」という回答が得られた．この結果から，DI の方がカスタマイズの記述時に好まれると言える．なお，すべての被験者が DI を選んだため，DI と BI の間の達成時間の比較は行えなかった．

アンケート結果の箱ひげ図（四分位数）を図 4.11 に示す．Q1 から Q6 は Iv Studio 全体に関する項目，その他は個別の機能に対する項目である．BI に対する評価項目である Q10 以外の項目に対し，中央値で 1 以上の評価を得た．特に，Q7 と Q11 は中央値 2 と好評であった．また，支援機能の型キャストと関数接続も高い評価を得た．その他のコメントとしては「インタラクティブなツールが簡単に作れて便利」のような好意的な回答が得られた．総合評価として，DI を用いた Iv Studio の機能は好評であった．

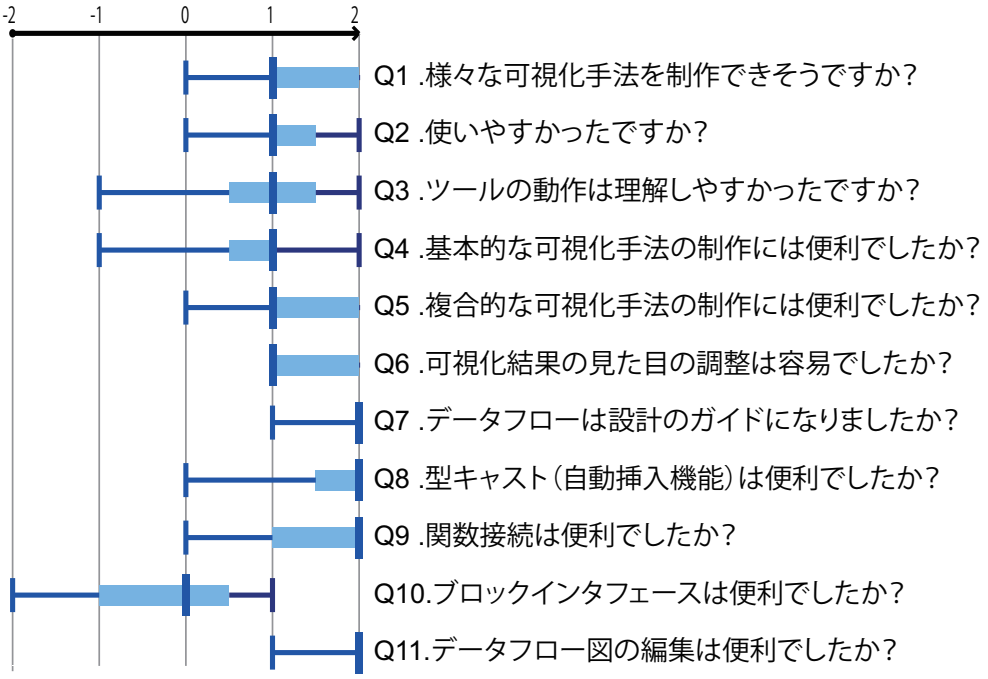


図 4.11: アンケート結果

4.5.4 実験 2

実験 1 の結果，タスク 3 においてすべての被験者が DI を選んだため，カスタマイズするタスクにおける達成時間の比較が行えなかった．そこで，そのような達成時間の比較を行うために，追加の実験を行った（調査項目 4）．

まず，実験 1 と同様に，はじめに操作方法を解説するチュートリアルを再生し，その後，画面の指示通りに操作するトレーニングモードを行ってもらった．その後，実験 1 のタスク 1 と同じタスクを練習として 1 回（タスク 4），復習としてもう 1 回（タスク 5）の合計 2 回，連続で行ってもらい，タスク 5 のみ達成時間を計測した．そして，2 分間の休憩を挟み，作成し

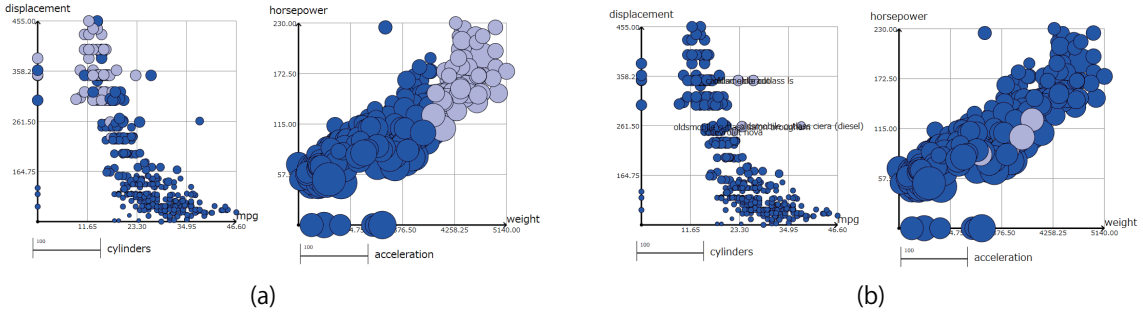


図 4.12: 実験 2 にてタスクの説明時に被験者に提示した図

た可視化手法を図 4.12(a) のようにカスタマイズしてもらうタスク（タスク 6）を行ってもらい，タスクの達成時間を計測した．さらに，連続して，タスク 6 にてカスタマイズした手法を，図 4.12(b) のようにカスタマイズしてもらうタスク（タスク 7）を行ってもらい，その達成時間を計測した．実験 1 と同様に，タスクは，単にバブルチャートや棒グラフを完成させることであり，図の配置や配色を厳密に同じにする必要はないと伝え，タスクをできるだけ速く達成するように求めた．そして，被験者がタスクを達成したと判断した時点で，その旨を被験者に宣言をしてもらった．実験者は，必要なノードが生成され，それらが適切に接続され，ノードのパラメータが適切に設定された状態をタスク達成とみなした．タスク未達成にもかかわらず被験者が終了を宣言した場合は，実験者から未達成の旨を伝えるようにした．実験には，27 インチ（解像度 2560 × 1600）のディスプレイを用い，全被験者が同じ PC を使用した．ただし，実験 1 と異なり，被験者はすべてのタスクを，DI か BI のどちらか一方のユーザインタフェース用いて行った．この理由は，カスタマイズの記述においては，記述の修正箇所の把握の速さも重要なため，初めて行うカスタマイズの記述の達成時間同士を比較する必要があったからである．また，タスク 4 では，実験 1 のように理解のしやすさを調査することが目的ではなく，ユーザインタフェースの使い方を覚えてもらうことが目的なことから，あらゆる質問を受け付け，被験者が戸惑った場合には実験者が支援するようにした．そのため，タスク 4 は時間の計測を行わなかった．その一方で，タスク 5 以降は，達成時間を比較するために，一切の質問を受け付けず，操作説明の資料も見られないようにした．また，BI については，DI と比較した際のマウスの移動量の影響が少なくなるように，左右 400 ピクセルずつ操作領域を狭め，1760 × 1600 の領域で操作できるように変更した．

タスク 6 では，横軸のデータ割り当てを Year から Weight に切り替え，右図をバブルチャートにカスタマイズしてもらった（図 4.12(a)）．タスク 7 では，左図に，選択したプロットの上に車種名のラベルを表示するようにカスタマイズしてもらった（図 4.12(b)）．ラベルを表示するための文字列のノードの作り方と使い方については，タスク 7 の前に説明し，その資料を渡してからタスクを行ってもらった．

被験者は，情報科学を専攻する学生及び大学院生（8 名）とソフトウェア開発企業に務める技術者（1 名）の合計 9 名で，全員にプログラミング経験があった．また，大学院生のうち 7 名が，情報可視化を専門とする研究室に 1 年以上在籍している学生であった．実験 2 の被験者には，実験 1 と同じ被験者が 3 名含まれていたが，実験 1 から 1 年以上時間が空いたため，影響は少ないと考えられる．4 名の被験者が DI を，5 名の被験者が BI を使いタスクを行った．

4.5.5 実験 2 の結果

図 4.9 に，タスク 5 の被験者ごとの達成時間を示す．達成時間の中央値を求めたところ，DI が 75.272 秒，BI が 129.740 秒であった．実験 1 と同様に，分布の仮定が困難なことから，ウィルコクソンの順位和検定を行ったところ，有意水準 5%で有意差が認められた (T = 11)．この結果から，BI の操作領域を狭めた場合でも，実験 1 のタスク 2 と同様に，DI の方が可視化手法を速く記述できると言える．

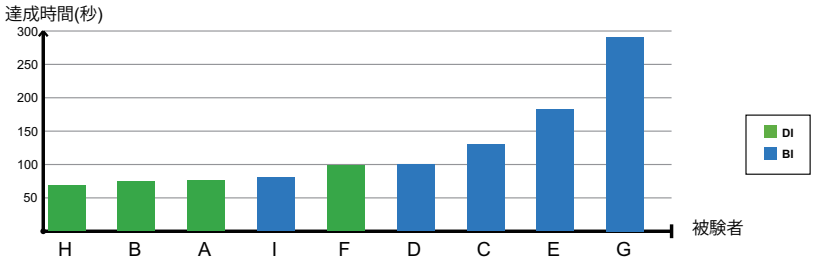


図 4.13: タスク 5 の被験者ごとの達成時間

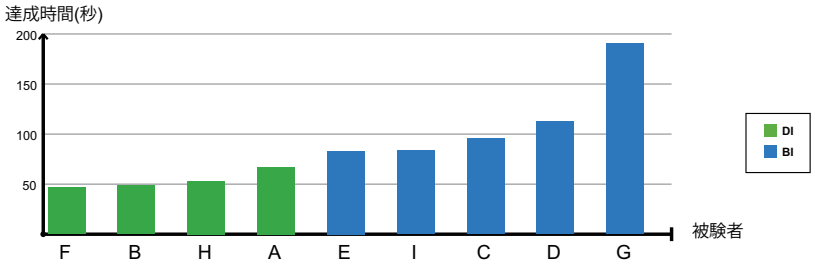


図 4.14: タスク 6 の被験者ごとの達成時間

図 4.14 に，タスク 6 の被験者ごとの達成時間を示す．達成時間の中央値を求めたところ，DI が 50.885 秒，BI が 95.772 秒であった．ウィルコクソンの順位和検定を行ったところ，有意水準 5%で有意差が認められた (T = 10)．タスク 6 の結果から，複数の修正を伴うカスタマイズを行うタスクにおいて，DI の方が可視化手法を速く記述できると言える．

図 4.15 に，タスク 7 の被験者ごとの達成時間を示す．なお，タスク 7 では，1 名の被験者の実験中に Iv Studio がクラッシュし，時間を正しく計測できなかったことから，8 名の比較を行った．達成時間の中央値を求めたところ，DI が 50.257 秒，BI が 114.717 秒であった．しかし，ウィルコクソンの順位和検定を行ったところ，有意水準 5%で有意差が認められなかった (T = 12)．

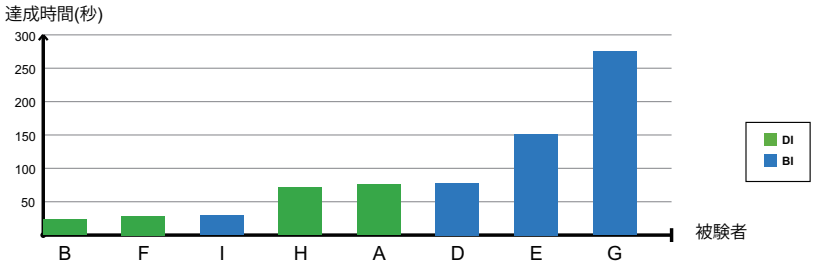


図 4.15: タスク 7 の被験者ごとの達成時間

4.5.6 実験の考察

タスク 1 の結果から，初めて利用する場合において，DI の方が可視化手法を速く記述できることが分かった．このタスクでは，どちらのユーザインタフェースかに関わらず，資料を参照しながら操作を行う被験者が多かった．そのため，この時間差は，被験者が練習で理解しきれなかった部分について資料を参照していた時間が，BI の方が長かったことが主な要因と考えられる．資料の総ページ数は，34 ページとどちらのユーザインタフェースも同じであった．この結果から，DI は BI と比べて理解しやすいユーザインタフェースと言える．

タスク 2 の結果から，DI の方が速く記述できることが分かった．効率性の向上は，以下に述べるように，ユーザインタフェースが備える 2 つの特徴と，記述中の手順を視覚的に確認できるという特徴による効果だと考えられる．

ユーザインタフェースの特徴の一つとして，DI ではディスプレイ上の狭い範囲で操作を完結できるということが挙げられる．それに対して，BI は Lyra の画面レイアウトをもとに設計を行ったものの，参考にしたレイアウトではピンの接続時にマウスカーソルをディスプレイの端から端まで移動させなければならず，操作に時間を要することとなった．BI が DI と比べてピンの接続操作にどの程度時間を要したかを被験者ごとに計算したところ，中央値は 6.728 秒であった．タスク 2 では，大画面かつ高解像度のディスプレイを用いたため，差が顕著に現れたと考えられることから，BI の領域を狭めたタスク 5 においても時間の比較を行ったが，同様に有意差が認められた．

もう一つのユーザインタフェースの特徴として，DI ではピンの接続先ノードの探索が容易なことが挙げられる．BI では，ピンの接続の際に，接続先のノードをノードパネルやキャンバスから探して選択する必要がある．タスク中には，目的のノードを探しまわるような被験者の仕草が観察され，タスク後に「ブロックパネルの内容が頻繁に切り替わり使いづらい」，「選択している間に何をしていたか忘れてしまった」とコメントする被験者もいた．このことから，DI は，接続先ノードの探索に要する操作量が少ないだけではなく，心理的な負担も軽減していたと言える．

記述中のデータのフローを視覚的に確認できるという特徴によって，現在の作業を確認する時間，あるいは次の手順を考える時間が短縮されたと考えている．このことは，定量的な測定は困難なものの，アンケート項目 Q7（データフローは設計のガイドになりましたか？）の結果と「データフロー図の完成形を思い描きながら作業に取り組んだ」というコメントが得られたことから推測される．

しかしながら,「データフロー図の整形が面倒だった」と回答した者もいた．また，1 名の被験者から DI に関して,「パラメータパネルが画面右端にあることが不便」というコメントも得られた．

タスク 3 ですべての被験者が DI を選んだことや，タスク後に得られたコメントの内容から，DI は手法のカスタマイズ作業において好まれることが分かった．また，タスク 6 において，データの割り当ての修正やマークの修正のような複数のカスタマイズを行う際には，DI の方が速く記述できることが分かった．一方，タスク 7 のような，ラベルを付けるだけのカスタマイズでは，有意差が認められなかった．この理由としては，タスク 7 では，最小の作

業を行った場合，BI でもブロックパネルの内容を切り替える必要がなく，DI との作業時間に差がほとんどなかったことと，操作時間や修正箇所の把握よりも，文字列のノードの使い方の理解にかかった時間の方が影響が大きかったことが考えられる．タスク 7 では，ラベルの文字列を表示するためにデータソースからマークのノードにピンを直接つなげる操作が必要であったが，タスク 6 以前にはそのような操作なかったため，被験者は，そのような操作に気が付くまでに時間を要していたと考えられる．

実験のアンケートでは，Iv Studio の機能は概ね好評であったものの，Q3（ツールの動作は理解しやすかったですか？）と Q4（基本的な可視化手法の制作には便利でしたか？）の評価は他の項目に比べて低かった．Q3 が低い理由は，「BI は使いづらい」という評価が混ざっているためである．Q4 が低い理由は，低い評価値をつけた被験者が「Excel では 1 クリックで作ってくれるから」と回答していたことから，散布図や棒グラフのような基本的な可視化手法を記述するだけならば，データフロー図の記述は面倒と感じたためと考えられる．ただし，Excel 等の表計算ソフトウェアにはない特長として，Iv Studio はテンプレートに用意されていない様々な可視化手法を記述できるという点を改めて強調しておきたい．

以上より，目標 3 に関して，ユーザインタフェースの使いやすさの向上が確認できた．ただし，この実験により示したのは，構築部品の対応関係の一覧が見えず，対応関係の記述にパネルの切り替えが必要なユーザインタフェースに対する改善であり，Lyra そのものとの比較ではないことは述べておく．

実験を通して分かった課題については，次のような改善を検討している．まず，データフロー図の整形が面倒という問題点については，データフロー図の自動整形機能などを設けることを考えている．次に，パラメータパネルが画面右端にあることが不便という問題点については，ノードをダブルクリックするとその横にパラメータパネルを表示するようにすることで改善できると考えられる．Q4 が低い，すなわち，基本的な可視化手法を記述する際の手間については，Iv Studio においても，基本的な可視化手法に関してテンプレートを追加することで改善できると考えられる．

4.6 データフロー図の描き方に関する調査

4.3.2 にて述べたように，マークのノードには，複合マーカーの記述を支援するために，入力された座標をそのまま出力するピンが備えられている．同様に，軸のノードにも，入力されたデータをそのまま出力するピンが備えられている．開発者は，これらのピンを利用して，例えば，棒グラフのラベルを表示する手法を記述できる．このような入力値をそのまま出力するピンのことをスルーピンと呼ぶ．スルーピンの効果を調査するため，Iv Studio を用いた可視化手法の記述を体験してもらった後に，アンケートに回答してもらう被験者実験を行った．

まず，被験者にある図を見せ，Iv Studio 上にあらかじめ用意されたノードをつなぐことで，その図を描くための手法を作成してもらい，Iv Studio による可視化手法の記述を体験してもらった．次に，データフロー図の描き方に関するアンケートを行った．調査方法としては，タスクを行ってもらい，そのタスクで描かれたデータフロー図を基準にすることも考えられた．

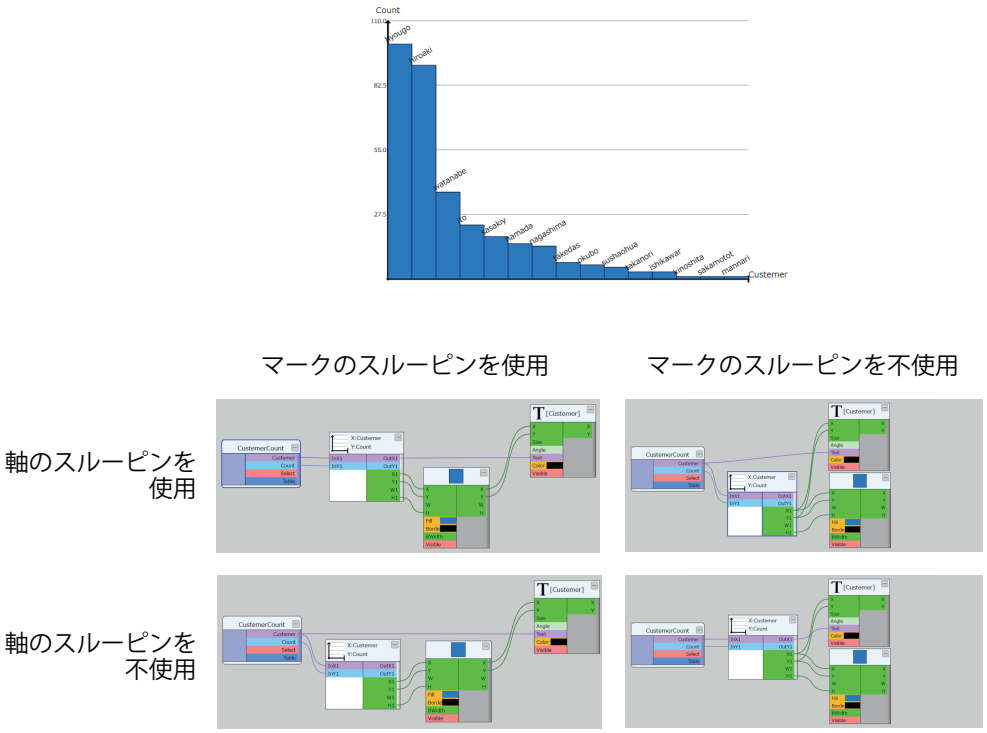


図 4.16: 質問 1 の可視化手法

しかし，スルーピンの使い方を詳細に説明した場合，そのピンを使うように誘導してしまう可能性や，逆に説明しない場合は使い方が分からずそのピンを使わない可能性が考えられた．そのため，Iv Studio による記述を体験してもらったうえで，改めてアンケートで全パターンを被験者に見せ，その中から選んでもらうような調査方法を選んだ．

アンケートでは，手法を記述する際のデータフロー図の描き方のパターンを見てもらい，次に同様の手法を作る場合，どの描き方を使いたいかを理由とともに回答してもらった．質問は 3 つあり，質問 1 は棒グラフの棒の上にラベルを付ける手法，質問 2 は散布図にラベルを付ける手法，質問 3 は 2 つの散布図を線でつないで表現する手法に関する質問であった．質問に用いた手法と，被験者に見せた描き方のパターンを，図 4.16 から 4.18 に示す．このうち，図 4.16 と図 4.18 は，アンケート前のタスクで実際に作成してもらったものである．

被験者は，情報科学を専攻する学生及び大学院生（10 名）とソフトウェア開発企業に務める技術者（1 名）の合計 11 名で，学生及び大学院生のうち 9 名が，情報可視化を専門とする研究室に 1 年以上在籍している学生であった．

表 4.5 に，アンケートの回答結果を示す．多くの被験者が，スルーピン使用を選択することを予想していたが，質問 1 と 2 に関しては，マークがおよそ半分，軸については不使用の方が多いという結果となった．一方，質問 3 に関しては，多くの被験者がスルーピン使用を選択した．

質問 1（棒グラフの棒の上にラベルを付ける手法）について，軸のスルーピン使用の方を

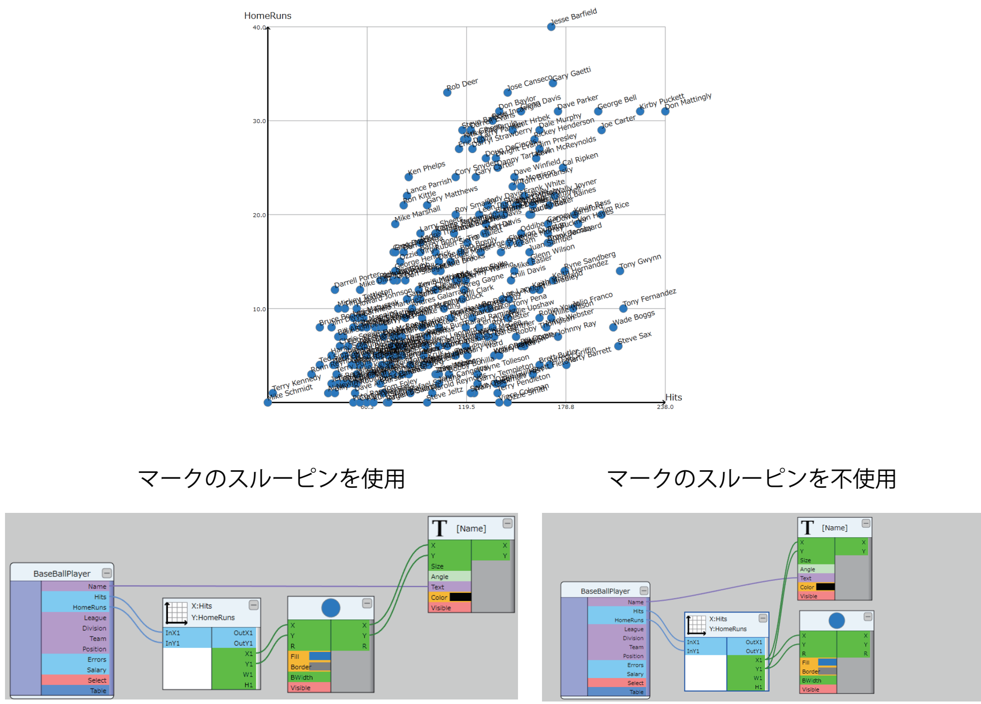


図 4.17: 質問 2 の可視化手法

表 4.5: データフロー図の描き方に関する調査の結果

	スルーピン使用	スルーピン不使用
質問 1:棒グラフにラベル付け（軸）	4	7
質問 1:棒グラフにラベル付け（マーク）	5	6
質問 2:散布図にラベル付け	6	5
質問 3:2 つの散布図を線でつなぐ	8	3

選んだ理由としては,「リンクが少ないほうがすっきり見える」といった意見が多かった．一方，不使用の方を選んだ理由としては,「ラベルはデータから直接つなげる方が直感的に分かりやすかった」といった意見が多かった．マークのスルーピン使用の方を選んだ理由については,「データフロー図がすっきり見えるようになる」という意見のほかにも,「ラベルの位置は棒グラフの位置となることが分かる」といったように，ラベルを棒グラフに関連付けて考えやすいという意見が多かった．不使用の方を選んだ理由については「同じ値は同じところから持ってきたい」「ラベルの位置は，座標変換したものを直接引っ張ってくるほうが正しいと感じた」「ラベルと矩形ノードは並列の概念であると考えた」といった意見があった．以上の結果から，同じ手法であっても，人によりマークの連携関係のとらえ方が異なることが分かった．また，ノードの連携関係を重視する人と，データフロー図の見やすさを重視する人がいることも分かった．

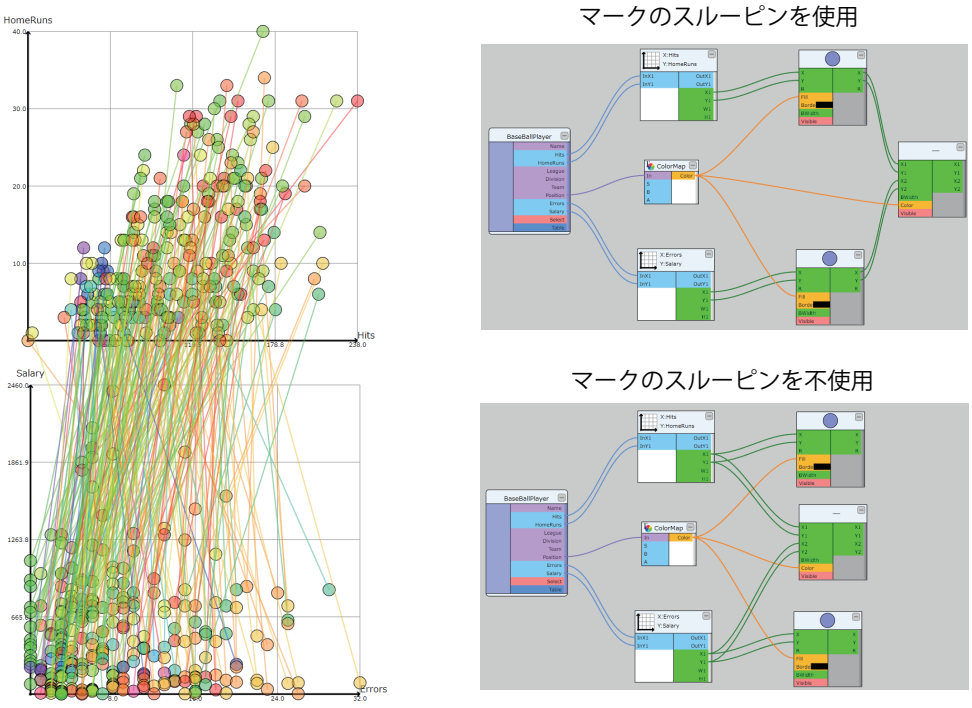


図 4.18: 質問 3 の可視化手法

質問 2（散布図にラベルを付ける手法）についても，質問 1 と同様の理由という被験者が多かったが，質問 1 ではマークのスルーピン不使用を選び，質問 2 では使用の方を選んだ者がいた．その被験者は，質問 1 では「パス長が短くノードを飛ばしていないから」という理由からマークのスルーピン不使用を選んでいたが，質問 2 では「散布図の個々のプロットとテキストの対応が分かりやすいと感じた」という理由からスルーピン使用を選んだ．そのほか，質問 2 でスルーピン不使用の方を選んだ被験者の中には，「ノードの形を変更したときに行う手間が少なくなりそうだから」「マークの変更が簡単になるから」という，質問 1 とは別の理由を回答した者もいた．以上の結果から，同じ人であっても，記述する手法によって重視することが異なることが分かった．

質問 3（2 つの散布図を線でつないで表現する手法）については，質問 1，2 と同様の理由からスルーピン不使用を選んだ被験者もいた一方で，さらに 2 名の被験者が，スルーピン使用を選んだ．その理由としては「丸と丸を線でつないでいることが分かるため」といった意見が多く，質問 3 の手法では，線が円と関連づいているイメージが強い被験者が多かったことが分かった．

調査の結果，同じ可視化手法でも開発者によって変換規則のイメージが異なることや，手法によっても変換規則のイメージが変わるということが分かった．また，処理のイメージを重視する人や，記述した後の修正のことを重視する人，データフロー図の見やすさを重視する人のように，開発者によってデータフロー図の記述で重視することが異なることも分かっ

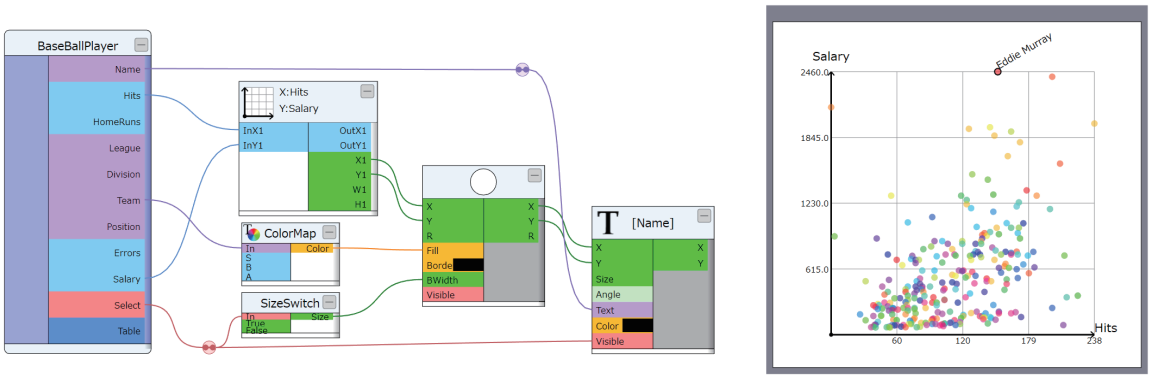


図 4.19: ベースボールプレイヤーのデータを可視化する手法の例

た．データフロービジュアル言語は，柔軟な記述を許すことから，このような好みの違いにも対応できる記述方法である．このことも，データフロービジュアル言語を可視化手法の記述のためのユーザインタフェースに用いる利点と言える．また，スルーピンのように，描き方の選択肢を増やし，開発者のイメージを損なわないようにすることも重要であることが分かった．この調査結果は，目標として設定した事柄とは逸れるものであるが，今後，情報可視化のビジュアルツールを開発するうえでの重要な知見である．

4.7　言語の表現力と記述量

目標 1 に関する評価として，開発したデータフロービジュアル言語による可視化手法の記述例を示す．まず，目標 1 にて設定した可視化手法の範囲に基づいて記述例を示す．次に，用意したノードの雛形の組み合わせによる変換方法の拡張の例として，ChronoView[SMT12] の記述を示す．さらに，典型的な可視化手法の記述例と，カスタイマイズ例を示し，それらに要する記述量に関して議論する．最後に，現段階では記述が困難な可視化手法について述べる．

4.7.1　基本的な可視化手法の記述

本項では，対象データ，表現形式，インタラクションの観点から，可視化手法の記述例を紹介し，目標 1 にて設定した範囲の可視化手法の記述が可能なことを示す．

対象データ

図 4.19 にベースボールプレイヤーの年収，ヒット数，所属チームのデータ [bbp] を可視化する手法の例を示す．この例では，X 軸でヒット数，Y 軸で年収，プロットの色で所属チームを表している．くわえて，選択されたプレイヤーのプロットの枠線を黒くハイライトし，そのプレイヤーの名前をプロットの上に表示する．このデータは，多次元データであり，年収と打率は量的データ，所属チームはカテゴリデータである．このように多次元データの取り

扱いが可能であり，データの種類については量的データとカテゴリデータの両種類の取り扱いが可能である．

図 4.20 に，グラフ構造の可視化手法の記述例を示す．図 4.20(a) は，同時に買われた商品間の関係を表すグラフ構造をノードリンクダイアグラムによって可視化した例である．商品は，その商品の売り上げ数と商品カテゴリの属性を持つ．商品を円，同時に買われた関係を線で表し，商品の売上数を円の大きさで，商品のカテゴリを色で表している．円の位置は，力指向アルゴリズムによって求められている．図 4.20(b) は，購入者と商品の関係を，2 層形式の連結図で表現したものである．購入者とその購入者が買った商品の間を線でつなぎ，線の太さで購入数を表している．このように，グラフ構造の取り扱いが可能であり，その頂点やエッジに重みがあった場合には，その重みの表現も可能である．

図 4.21 には，木構造の可視化手法の記述例を示す．いずれの図も，3D キャラクタモデルの関節の構造を可視化したものであり，関節に物理演算用の設定がされているか否かを色で表現している．図 4.21(a) は，ノードリンクダイアグラム，(b) はアイシクルツリーである．また，図 4.21(c) は，円周状のノードリンクダイアグラム，(d) は，サンバースト図である．木構造のレイアウトを行うノード TreeDiagram は，レイアウトした頂点の座標のほか，座標計算中に副次的に求められる，幅や高さ，角度の情報も出力する．それらを利用することで，開発者は，(b) や (d) のようなバリエーションも記述できる．このように，木構造の取り扱いが可能であり，その頂点に属性が付加されていた場合には，その属性を表現することも可能である．

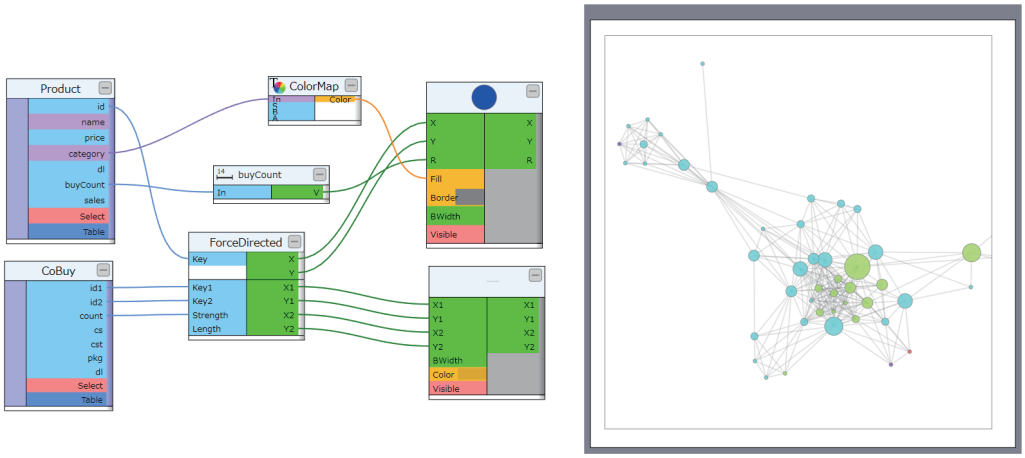
表現形式

散布図や棒グラフのような座標系の基本的な表現を記述できることは，図 4.2 や図 4.4，図 4.19 にて示した通りである．図 4.10，図 4.11 も Iv Studio を用いて作成した図である．図 4.22(a) は，時間帯ごとの商品の売上数を扇型の半径で表した図である．このような極座標系を用いた手法を記述することもできる．

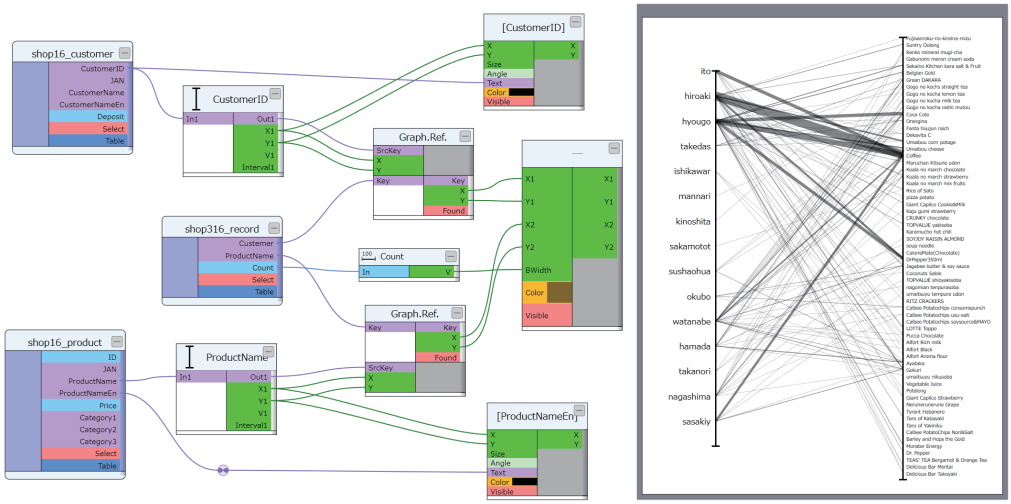
図 4.22(b) は，領域系の表現の例である．この図は，3 都市の気温のデータを，縦軸に気温，横軸に日付でプロットし，同じ都市の気温を凸包で囲った表現である．囲いの表現は，凸包の計算ノードと多角形のノードの 2 つのノードを用いて記述した．

図 4.22(c)(d) は，配列系の表現の例である．顧客毎の商品の購入数を表している．図 4.22(c) は，行列のマスの色を用いて購入数を表している．2 つの単軸のノード，値を色に変換するノード，マークのノードの合計 4 つのノードを用いて記述した．図 4.22(d) は，行列のマス内の棒の長さによって購入数を表現している．この他，円の大きさや棒の位置を用いて値を表現する手法 [PDF14] も容易に記述することができる．

連結系の表現については，図 4.20 や図 4.21 にて記述可能なことを示した．その他の例として，図 4.22(e)(f) を示す．図 4.22(e) は，自動車のデータを表した PCP である．各座標軸のための単軸のノード 7 つと折れ線のノードの合計 8 つのノードを使い記述した．図 4.22(f) は，左に自動車の燃費と馬力，右に年式と車重の散布図を描き，選択された自動車のみ，両散布図の対応する円を線で結んで表現する手法である．両散布図の円のノードの出力を線のノー

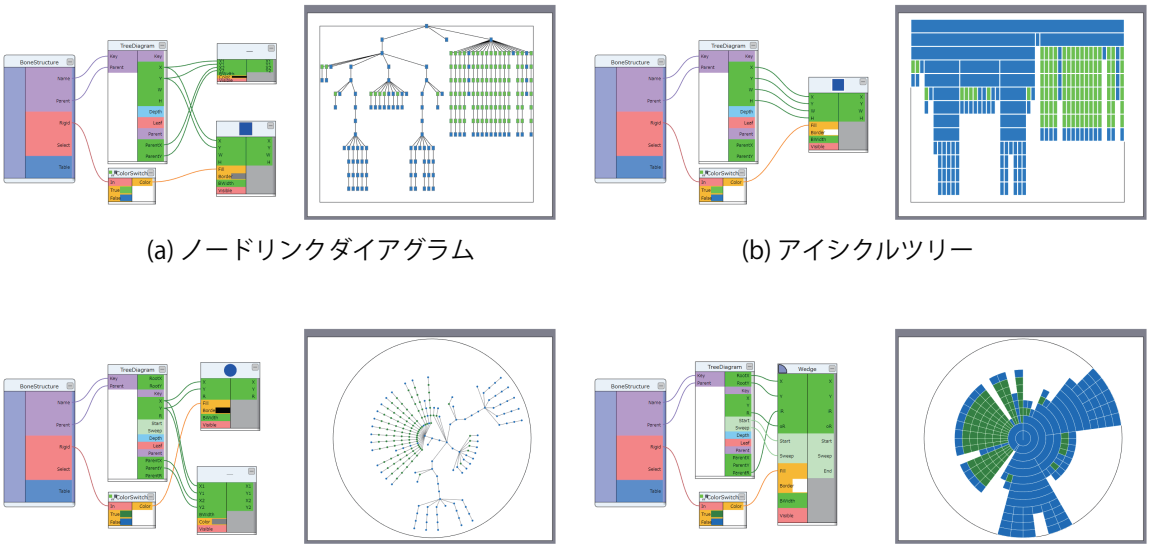


(a) ノードリンクダイアグラム



(b) 2 層形式の連結図

図 4.20: グラフ構造の可視化手法を記述した例



(a) ノードリンクダイアグラム　(b) アイシクルツリー

(c) ノードリンクダイアグラム（円周状）　(d) サンバースト図

図 4.21: 木構造の可視化手法を記述した例

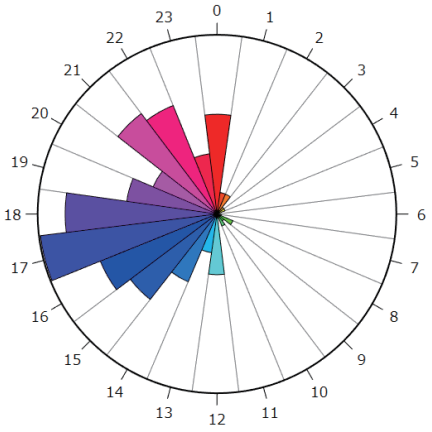
ドに接続することで記述した．

以上の通り，図表現の 4 つの基本系のそれぞれを，開発したデータフロービジュアル言語により記述することが可能である．このことは，Shneiderman ら [CMS99] の提案した可視化手法の構成要素の分類，Marks, Connection, Enclosure も記述できることを意味している．Marks で述べられている点や線，面はマークのノードとして用意されている．Connection と Enclosure による関係性の表現は，図言語の連結系と領域系の記述例で示したとおり記述することができる．

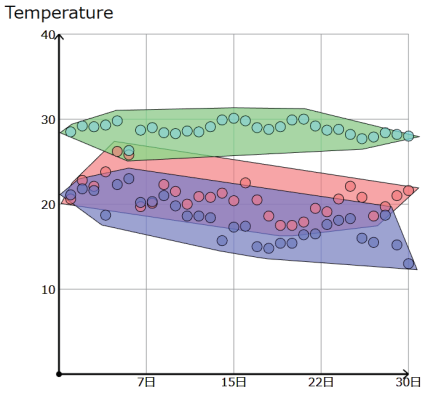
インタラクション

インタラクションを記述可能なことについては，図 4.2 や図 4.4，図 4.19 にて示した通りである．Linking & Brushing や，選択したレコードのみ詳細な情報を表示するようなインタラクションを記述できる．

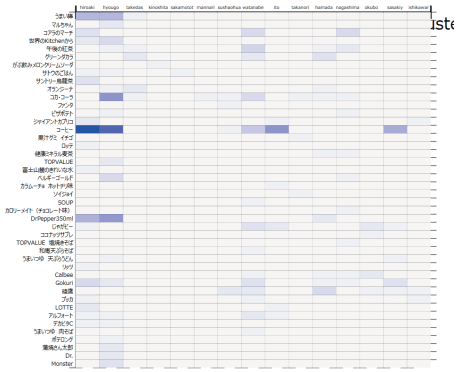
以上に示した通り，開発したデータフロービジュアル言語は，目標 1 で定めた範囲の可視化手法を記述することが可能である．また，これらの手法は，データテーブルのノードを除けば，いずれも 10 ノードに満たないノード数で記述可能である．重要なことは，開発したデータフロービジュアル言語では，これらの可視化手法を，様々に組み合わせられるという点である．このことが，情報可視化を利用する際に必要な，データや利用目的に合わせたカスタイマイズを可能にする．



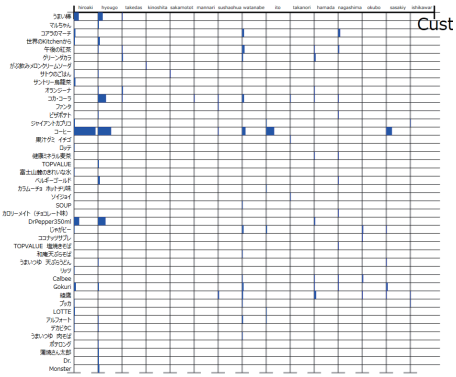
(a) 座標系の例(極座標系)



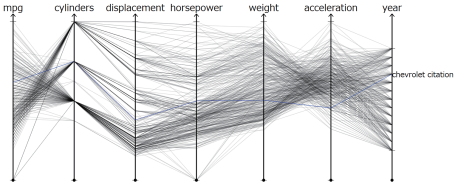
(b) 領域系の例(凸包による囲み表現)



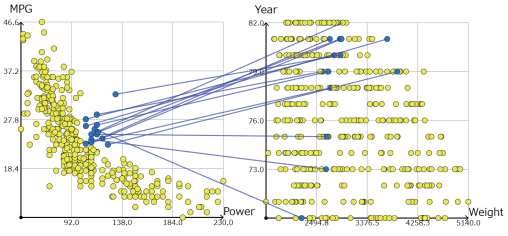
(c) 配列系の例1 (色の濃淡による表現)



(d) 配列系の例2 (棒の長さによる表現)



(e) 連結系の例1 (PCP)



(f) 連結系の例2 (2つの散布図間の対応関係を表現)

図 4.22: 基本的な表現系に基づいて記述した可視化手法の例

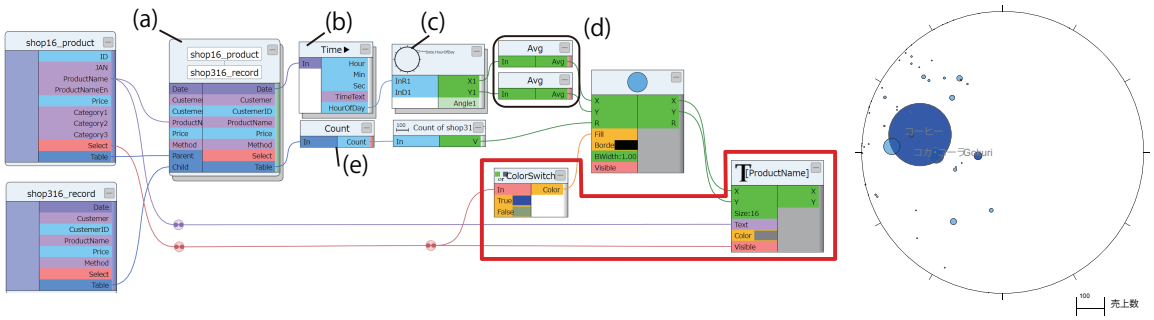


図 4.23: ChronoView を記述した例

4.7.2 ChronoView の記述

用意した変換方法のノードと，処理演算用のノードを組み合わせることで，複雑な可視化手法も記述することができる．その例として，ChronoView の記述を紹介する．

ChronoView は，タイムスタンプを持った多くのイベントグループを可視化する手法である．各タイムスタンプがアナログ時計を模した円周上に配置され，イベントグループがタイムスタンプの位置の重心に配置される．このように，ChronoView には，データの座標軸上へのマッピングと，その座標値の集約が必要となるため，前項で紹介した可視化手法と比べて記述が複雑となる．

図 4.23 は，購買データを可視化する ChronoView を記述した例である．購買データは，売上のテーブルと商品のテーブルを持つ．これらのテーブルから，ノード (a) を用いて商品毎に売上レコードを集計した．次に，ノード (b) を用いて商品の売上の日時の情報を時刻 (0 時から 23 時) に変換し，ノード (c) を用いて極座標軸上の座標を求めた．そして，平均を計算するノード (d) を用いて重心を計算し，重心を円の座標に割り当てた．最後に，ノード (e) を用いて商品の売上数を求め，円の半径に割り当てることで，ChronoView を記述した．くわえて，円を選択すると色を変更し，そこに商品名を表示するインタラクション手法も記述した (図 4.23 の赤枠内)．

この例のような，予め用意された変換方法と演算処理を組み合わせる記述は，1 つの変換方法とマークの組み合わせしか行えない既存のビジュアルツールでは記述できない．仮にこのような記述を可能にしたとしても，ユーザインタフェースがそのままではデータの流れを確認できないため，実際的な利用は難しいと考えられる．この例は，データフロービジュアル言語による記述が，既存のビジュアルツールよりも多様な可視化手法を記述可能にしたことを示す一例である．

ChronoView の記述については，プログラミング言語との比較も行った．情報可視化のプログラム制作に慣れた 3 名の被験者に，それぞれ得意なプログラミング言語を用いて ChronoView を記述してもらった．被験者は全員，ChronoView の制作経験があった．

記述してもらったコードから，IDE で生成されたコードやデータの読み込みのためのコードを除き，行数を数えた．その結果，C#で 74 行，Processing で 110 行，JavaScript+D3 で 65

表 4.6: 典型的な可視化手法の記述に要したノード数

可視化手法	ノード数
散布図	2
バブルチャート	3
棒グラフ	2
折れ線グラフ	4
面グラフ	7
PCP（7 次元）	8
ノードリンクダイアグラム（グラフ構造）	3
行列表現	4
ノードリンクダイアグラム（木構造）	3
アイシクルツリー	2

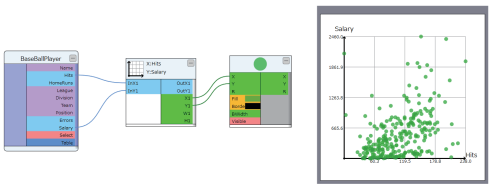
行であった．制作に要した時間は，C#を用いた人が約 20 分，Processing を用いた人が約 45 分，D3 を用いた人が約 100 分であった．これに対し，Iv Studio では，インタラクション部分を除いて 10 ノード，5 分程度で ChronoView を記述できた．

4.7.3 記述量に関する議論

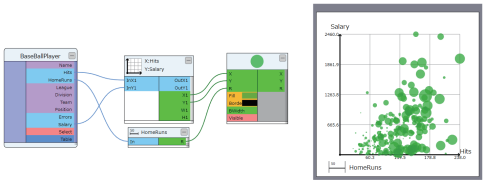
可視化手法の記述に要する記述量について議論する．本項では，記述量の指標として，ノード数を用い，それに基づいた議論を述べる．

まず，典型的な可視化手法の記述例を図 4.24 に示す．本論文では，これまでにも可視化手法の記述例を示してきたが，ノード数に関する議論を行うために，最もシンプルな例を改めて示す．多次元データについては，一般的な手法として，散布図（図 4.24(a)），バブルチャート（図 4.24(b)），棒グラフ（図 4.24(c)），折れ線グラフ（図 4.24(d)），面グラフ（図 4.24(e)）の記述例を示す．また，多次元データの可視化でよく用いられる PCP（図 4.24(f)）の記述例も示す．グラフ構造については，ノードリンクダイアグラム（図 4.24(g)）と行列表現（図 4.24(h)）の記述例を示す．木構造については，ノードリンクダイアグラム（図 4.24(i)）とアイシクルツリー（図 4.24(j)）の記述例を示す．表 4.6 に，これらの可視化手法の記述に要したノード数を示す．表中のノード数は，データテーブルのノードを除いたものである．

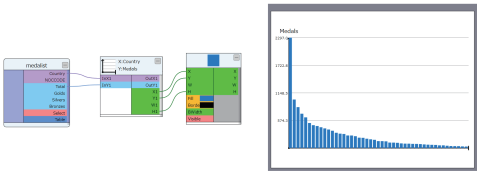
ここで示した大半の手法が，変換方法とマークのノードの組み合わせのみで記述することができ，その記述に要したノード数は 2 から 4 個である．このことから，開発した言語は，多くの典型的な可視化手法を簡潔に記述できると言える．しかし，折れ線グラフと面グラフについては，変換方法とマークのノード以外に，演算処理のノードを要した．開発したデータフロービジュアル言語では，基本ルールとして，データテーブルの持つレコード数分のマークが生成されるが，折れ線グラフと面グラフは，1 つの折れ線や多角形で十分である．このような場合に対応するため，開発した言語では，レコード方向のデータを 1 つの配列オブジェク



(a) 散布図

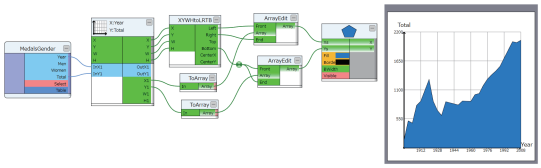


(b) バブルチャート

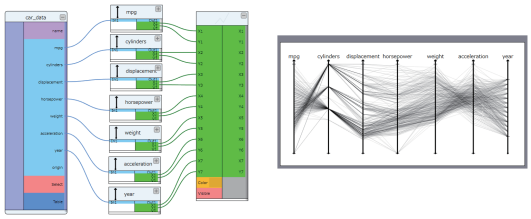


(c) 棒グラフ

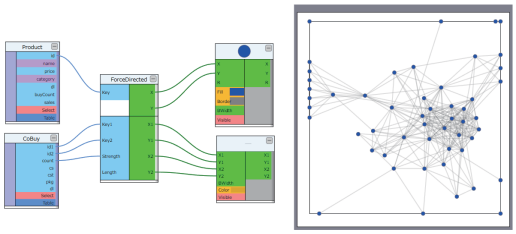
(d) 折れ線グラフ



(e) 面グラフ

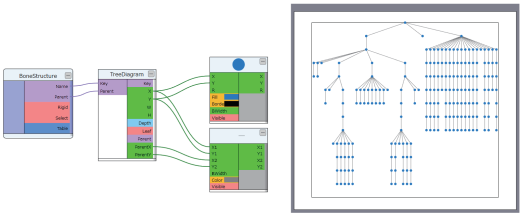


(f)PCP

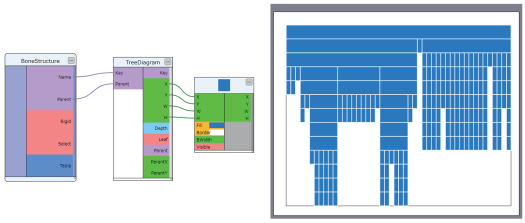


(g) ノードリンクダイアグラム（グラフ構造）

(h) 行列表現



(i) ノードリンクダイアグラム（木構造）



(j) アイシクルツリー

図 4.24: 典型的な可視化手法の記述例

トに変換するノードの雛形が用意されており，開発者がその雛形を利用して折れ線グラフ等を記述できるようにしている．この変換処理が必要なため，棒グラフ等と比べて記述にかかるノード数が多くなる．ただし，折れ線グラフについては，操作量の面からみると，通常の数値型のピンと配列の数値型のピンをつないだ場合には，型キャスト機能が働くため，記述にかかる操作量は棒グラフや散布図と変わらない．面グラフについては，配列化の処理に加えて，直交座標軸の端の点を配列の前後に加える処理が必要となるため，さらに多くのノードを要する．ノードの雛形の設計方針として，マークには処理機能を持たせないようにしたが，面グラフについては，汎用的な演算処理だけでは記述が複雑となるため，今後，記述量を減らすための工夫が必要と考えられる．折れ線グラフや面グラフ以外に，PCP の記述にも多くのノードを要した．開発したデータフロービジュアル言語には，平行座標系を雛形として用意していなかったため，1 カラムずつ軸を用意し，その出力を折れ線へ入力していくという記述が必要となった．雛形が用意されていなくても，単軸を利用して雛形にない手法を記述できるということが，開発した言語の利点ではあるものの，簡潔に記述するという観点からは今後の課題と言える．

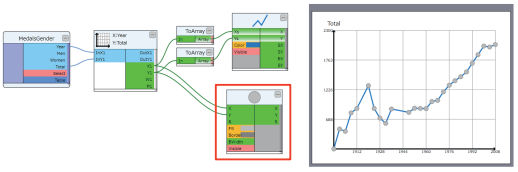
次に，カスタマイズの例を図 4.25 に示す．図中の赤枠内が，カスタマイズのために追加されたノードである．マークを付加するカスタマイズとして，ラベル表示（図 4.25(a)），数値の表示（図 4.25(b)），折れ線への点の付加（図 4.25(c)）の記述例を示す．(a) と (b) の図は共に，オリンピックの国ごとのメダル獲得数 [med] を可視化した棒グラフのカスタマイズ例であるが，(a) は国名を表示，(b) は獲得したメダル数を表示するという違いがある．マークの使われていない視覚的属性に，新たなデータを割り当てるカスタマイズとして，色によるデータ提示の追加（図 4.25(d)），大きさによるデータ提示の追加（図 4.25(e)）の記述例を示す．最後に，近年の情報可視化でよく用いられるマルチプルビューを想定したビュー間の連携のカスタマイズとして，線でつなぐ表現（図 4.25(f)），色によるハイライト表現（図 4.25(g)），枠線のハイライト表現（図 4.25(h)）の記述例を示す．このように，この程度の基本的なカスタマイズであれば，1 から 2 個のノードで簡潔に記述することができる．

以上のシンプルな状態の手法の記述にかかるノード数を踏まえたうえで，実用面を考えた場合のノード数について考察する．まず，典型的な可視化手法に対してカスタマイズを加えたような手法のノード数は，多くとも 10 程度と考えられる．例として挙げたようなカスタマイズを行った場合，視覚的な要素が増えることから，カスタマイズを加えるほど視覚的な混雑度が増す．そのため，行えるカスタマイズは 3 から 4 個程度が限度と考えられ，2 から 4 ノード程度で最もシンプルな状態が記述できる手法にカスタマイズを加えた場合には，ノード数は，多くとも 10 程度になると考えられる．PCP や面グラフのような，シンプルな状態の記述に多くのノードを要する手法を用いた場合には，それ以上となる可能性はあるが，その場合も，およそ 15 程度と考えられる．

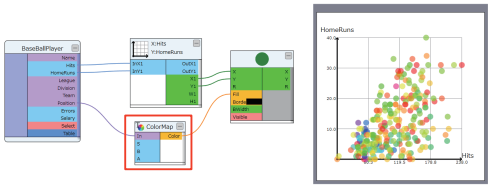
次に，マルチプルビューを用いた場合のノード数について考察する．実システムで利用されるビュー数について正確に想定することは困難であるが，情報可視化に関する国際会議 IEEE Visual Analytics Science and Technology (VAST) で発表される可視化ツールの多くが，3 から 5 程度のビューを持っていることから，その数を指標とする．VAST で発表される可視化ツール

(a) ラベル表示

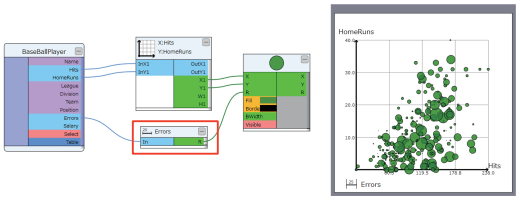
(b) 数値表示



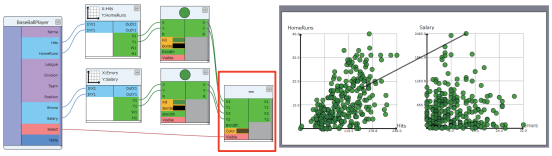
(c) 折れ線への点の付加



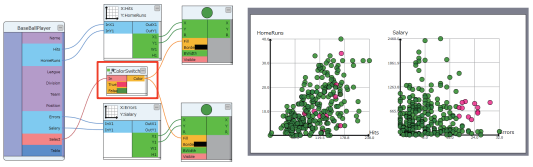
(d) 色によるデータ提示の追加



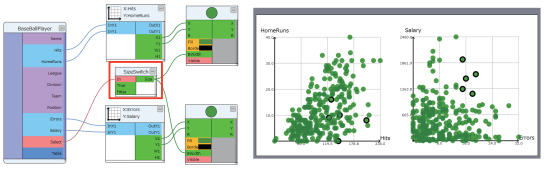
(e) 大きさによるデータ提示の追加



(f) 線でつなぐ表現



(g) 色によるハイライト表現



(h) 枠線のハイライト表現

図 4.25: 可視化手法のカスタマイズ例

のビューのうち一つは，レコードの詳細な情報を提示するために使われることが多く，そのようなビューの記述は本研究の対象外である．このことから，おおよそ 4 ビュー程度の記述が行えれば，十分に実用的と考えられる．その観点から，4 ビューの可視化手法にかかるノード数を考えると，先に示した典型的な可視化手法やカスタマイズにかかるノード数のおよそ 4 倍程度であり，多くとも 40 ノード程度である．

最後に，データフロー図の複雑性について考察する．例えば 4 ビューの可視化手法を記述した場合，40 ノードが密接に接続されるわけではなく，各ビューがある程度独立したデータフロー図となる．そして，各ビューのためのデータフロー図をまたぐようなノードは，データフロー図の始点であるデータソース部分や，連携のための変換方法やマークのみである．そのため，データフロー図の複雑さとしては，開発者の取り扱いが可能な程度と考えられる．しかしながら，どの程度の複雑さまでなら取り扱いが可能なのかや，容易に記述が可能なのかについては，被験者実験を通して検証すべき今後の課題である．また，ノードのグループ化など，マルチプルビューの記述を想定した支援機能についても，今後検討が必要である．その一方で，マルチプルビューに関しては，開発者は，次章で述べる埋め込み可能な実行環境を利用して，埋め込まれる先のプログラムによりビューを管理することもできるため，大規模な可視化のシステムを構築する場合には，1 つのデータフロー図で記述することにこだわる必要はない．開発者にすべてをデータフロービジュアル言語を用いて記述することを強いるのではなく，開発者が管理しやすい形を選択できるようにしたことが，Iv Studio の開発環境としての大きな利点であることを述べておく．

4.7.4 制限

Bertin[Ber84] の提唱した 8 つの視覚変数のうち，形ときめの記述については，直接的には対応していない．条件分岐等のノードを駆使することで記述することはできるが，データフロー図が複雑となり，あまり現実的ではない．ただし，これらの視覚変数に対応したノードの雛形を追加することで容易に対応可能である．きめに関しては，一般的な描画 API のブラシに相当する概念を導入することで，より発展的な対応が可能になると考えている．

開発したデータフロービジュアル言語は，レコード単位の処理を基本としている．そのため，隣のレコードの値を参照するような手法を記述できない．例えば，一つのカラムについてレコード毎にデータを積み上げていくような積み上げ棒グラフを記述できない．これに対しては，一つ前のレコードの値を取得するといったノードを加えることで対応可能と考えている．

現在の実装で記述可能なインタラクション手法は，図形の選択のみであるが，今後さらに拡張が可能と考えている．例えば，スライダーバーのような GUI コントロールのノードを追加することで，より多様なインタラクション手法の記述が可能になると考えている．ただし，Iv Studio という開発環境として考えた場合，次章で述べる埋め込み可能な実行環境の仕組みにより，可視化処理を埋め込まれた先のプログラムとの連携が可能であるため，データフロービジュアル言語上で GUI コントロールの記述まで行えるようにする利点については議論の余地がある．

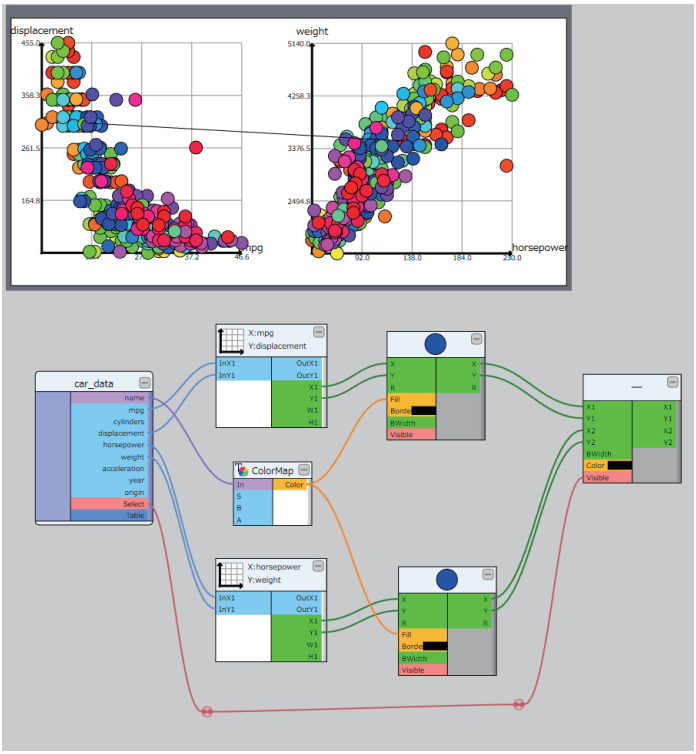


図 4.26: Iv Studio の使い方の資料にて題材とした可視化手法のデータフロー図

4.8 利用事例

Iv Studio を，移動体の軌跡のデータ分析に関する研究に利用してもらった．利用者は，情報科学を専攻する大学生で，移動体の軌跡のデータのクラスタリングに用いる特徴ベクトルに関する研究を行っていた．その学生は，複数の種類の特徴ベクトルを考案しており，その特徴ベクトルの比較のために，可視化を用いたいというモチベーションがあった．既に Java で作成したプログラムを使って可視化を行っていたが，そのプログラムでは 1 種類の特徴ベクトルに関する可視化しか行うことができず，複数の比較に対応するためには大幅な改造が必要であった．そこで，複数の比較を行う可視化プログラムの制作に Iv Studio を利用してもらった．なお，この学生は，Iv Studio の使用経験はなく，他のデータフロービジュアル言語の使用経験もなかった．筆者は，その学生に，Iv Studio が読み込み可能なデータ形式と Iv Studio の利用方法を 5 分程度説明してから，Iv Studio の使い方の資料を渡し，その後，自由に利用してもらった．使い方の資料では，図 4.26 に示す可視化手法の記述を題材とした．

学生が作成した可視化手法とそのデータフロー図を図 4.27 に示す．この可視化手法では，特徴ベクトルについて，多次元尺度構成法によって二次元に圧縮した結果と k-means 法によりクラスタリングした結果を保存したデータテーブルが 2 つ（2 種類の特徴ベクトル分）読み込まれており，多次元尺度構成法によって圧縮された結果がそれぞれ，左右に並べた散布図として表現される．クラスタリング結果は，散布図のプロットの色を用いて表現される．く

わえて，片方の散布図でプロットが選択されると，両散布図間のプロットの対応関係が，選択した方と反対側のプロットと同じ色の線で結んで表現される．

作成途中，渡した資料に説明のなかった他テーブルの参照のためのノード（図 4.27(a)）の使い方について，筆者がサポートしたものの，それ以外のサポートは必要とせずに，その学生は手法を作成することができた．関係演算子（図 4.27(b)）の使い方については説明しなかったが，学生が自分でその機能を見つけて利用していた．学生は，その機能を，片方の図のクラスタすべてから他方に線を引くために利用していた．図 4.27 の可視化手法は，手法の設計の試行錯誤や，データ分析の時間を含めて，およそ 1 時間半程度で作成できた．

数分の説明と手法の作成方法の説明資料によって，その学生は，Iv Studio を使って目的とする可視化手法を作成することができた．既に作成していた Java のプログラムを改造する場合と比べた感想を聞いたところ「現在の私のツールでは 2 つの図を表示することを想定していないため，新たに実装が必要です．さらに 2 図をリンクさせるインタラクションを新たに実装しなくてはいけないため実装コストとは別に勉強するコストもかかっていたと思います．そのため約 1 時間半で目的の図が作成できるのはとても便利であると感じました.」と回答を得た．その学生は，その後，図 4.27 の可視化手法の他にも，移動体の向きをマークの角度で表す可視化手法を作成するなど，別の手法の記述にも Iv Studio を利用していた．

関係演算子を用いた表現の作成から，その学生は単に説明資料に従って手法を作成したのではなく，Iv Studio の使い方を理解し，目的の手法を作り上げていたと考えられる．別の視点から，この利用事例は，実利用を想定した際には汎用的な演算機能を持たせることが重要であることを示す例の一つと言える．汎用的な演算機能を持たせることで，足りない機能を補うことができ，目的に合った使い方が可能となったと考えられる．

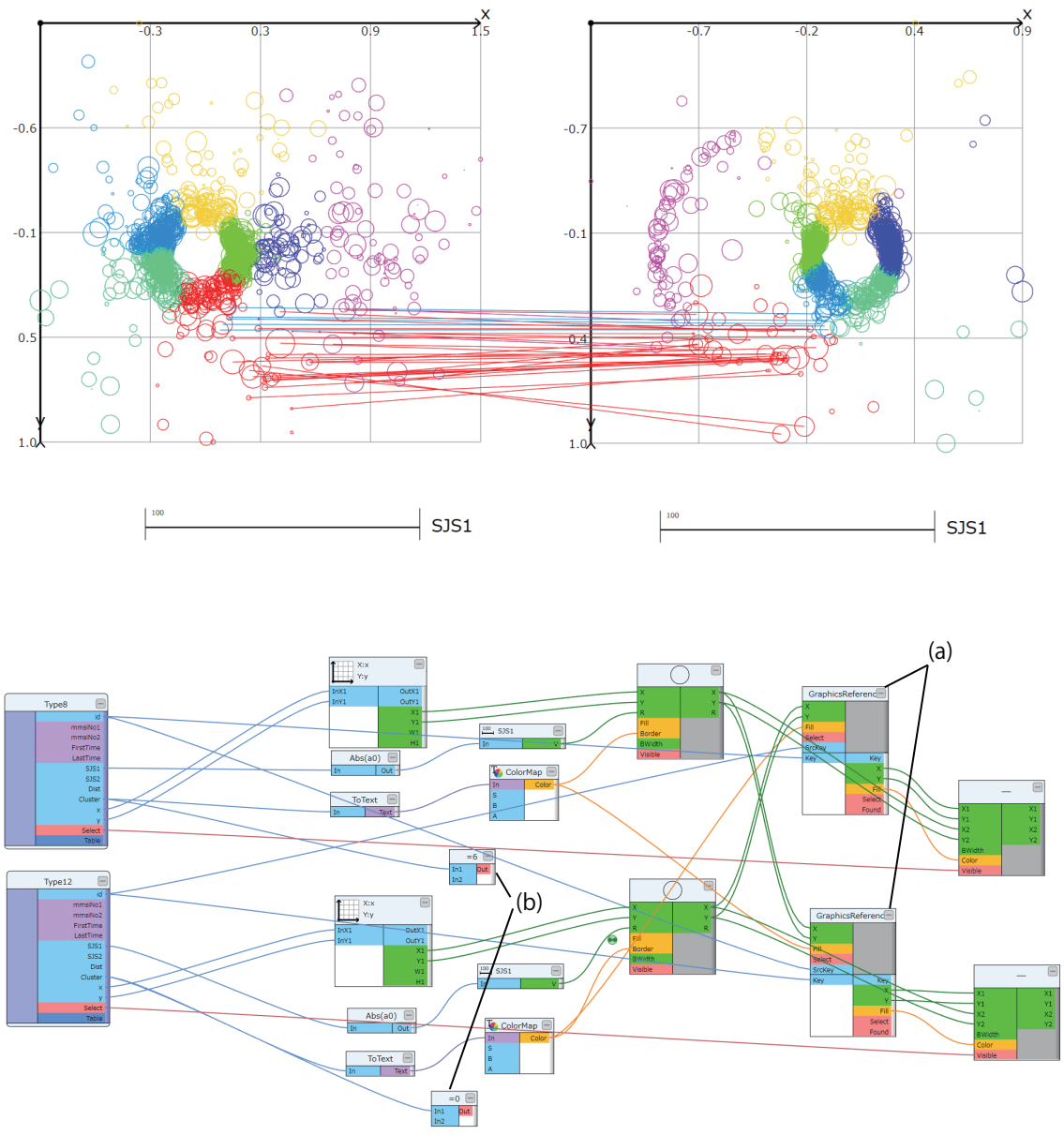


図 4.27: 特徴ベクトルを比較するために作成した可視化手法のデータフロー図

第5章　実システムへの埋め込みが容易な情報可視化の実行環境

本章では，目標 5 を達成するために設計した実システムへの埋め込みが容易な情報可視化の実行環境について述べる．また，この環境に関連して，ノードの雛形の拡張（目標 2）についても述べる．まず，実行環境の設計方針を述べる．次に，設計した可視化の実行環境とノードの雛形の拡張方法について説明する．その後，実行環境の実行パフォーマンスの評価を示し，最後に，利用事例を通して有用性を示す．

5.1　実行環境の設計方針

既存のビジュアルツールやデータフロービジュアル言語を用いたシステムでは，ツール上で実装した可視化処理を，ツール外から利用できるようにするためのランタイムライブラリ等が提供されている．しかしながら，この方法では実装した可視化処理の利用先が限定される．本研究では，幅広いプログラムから可視化処理を利用できるようにすることを目指す．

可視化を行うプログラムには，データの読み込みや生成した画像の表示が必要である．また，インタラクション手法を用いた場合は閲覧者からの入力の受付も必須である．本研究では，以下のような環境のプログラムにおいて，ツール上で実装した可視化処理を利用できるようにすることを目標とする．

- プログラミング言語
  - C（C++，Objective-C），Java，JavaScript，C#（共通言語基盤 [cli] 対応言語）
- データ形式
  - CSV，スプレッドシート，SQL，独自フォーマット，オンメモリ，etc.
- 描画 API
  - GDI，Processing，Java2D，Canvas，OpenGL，DirectX，etc.
- GUI システム（GUI ツールキット）
  - WIN32 API，Cocoa，X，Swing，Qt，etc.

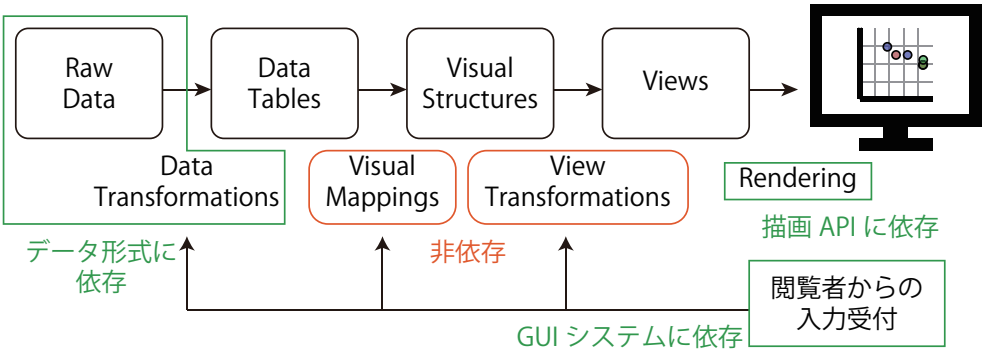


図 5.1: 依存／非依存部分の切り分け

このように，描画 API や GUI システムに目を広げると，考慮すべき環境が非常に多い．これらの環境全てを網羅するようにランタイムライブラリを開発することができれば，目標は達成できるが，現実的ではない．そこで，本研究では，以下の方針で実行環境の設計を行った．

1. 依存／非依存部分を切り分け
2. 非依存部分の実行環境を選定し，Iv Studio からの出力機能を開発
3. 実装が容易な依存-非依存部分間のインタフェース関数を設計

本研究では，Iv Studio の利用先を増やすことを目的としているが，本章で述べる実行環境の設計は，他のビジュアルツールにおいても有用な実装に関する知見である．

5.2 実行環境の設計

本節では，実行環境の設計について述べる．また，ノードの雛形の拡張方法についても説明する．

5.2.1 依存／非依存部分の切り分けと全体の設計

まず，情報可視化を行うシステムのうち，埋め込み先のプログラムの環境に依存する部分と非依存の部分の切り分けを行い，実行環境がカバーすべき範囲について検討した．この検討では，プログラムの環境として，データ形式，描画 API，GUI システムの 3 つを考慮した．

情報可視化のシステムは，情報可視化のリファレンスモデルの形を取ることが知られている．そこで，そのモデルに照らし合わせながら，データ形式，描画 API，GUI システムに依存する部分を検討した (図 5.1)．

Raw Data とは，保存されているデータのことのため，データ形式そのものである．Raw Data を Data Tables に変換する処理である Data Transformations は，データ形式に依存する処理であ

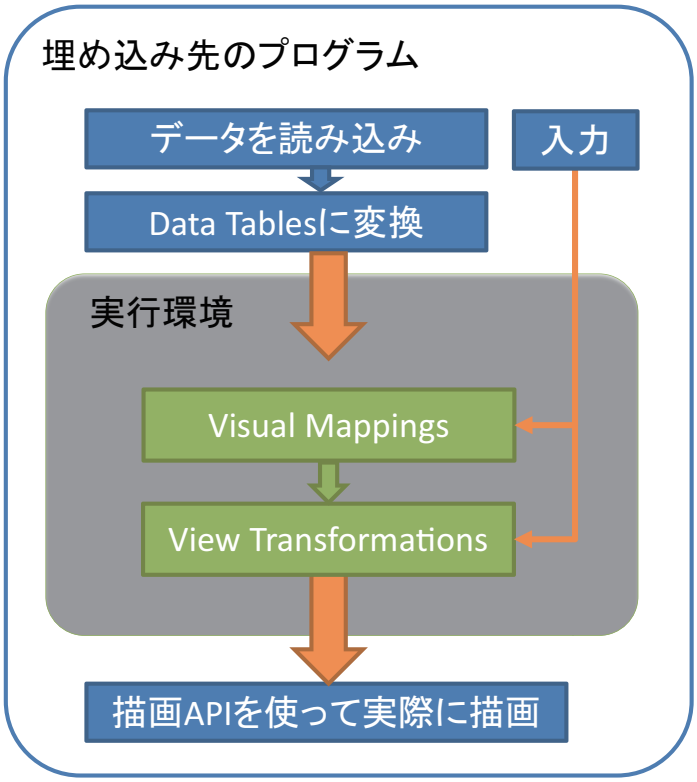


図 5.2: 可視化処理全体の実行イメージ

る．リファレンスモデルの後半に目を向けると，Views を画面に表示する Rendering は，埋め込み先のプログラムが使用する描画 API に依存する処理である．インタラクションについて考えると，閲覧者からの入力を受け付ける処理が，GUI システムに依存する．一方で，Visual Mappings と View Transformations は，純粋な演算処理であることから API 等に依存しない．以上のことから，Data Transformations までと，実際に画像を表示する Rendering，GUI システムからの入力受付処理は，埋め込み先のプログラムに任せ，実行環境は Visual Mappings と View Transformations を行うことが望ましいと考えられる．図 5.2 に本実行環境を用いた場合の可視化処理全体の実行イメージを示す．

この設計を用い，利用先として想定するプログラミング言語（以下,「想定言語」と呼ぶ）で動作し，かつ，埋め込みを容易にするために，以下の 2 つを行った．

- 想定言語への埋め込みが容易な非依存部分の実行環境の選定と，その環境に対する Iv Studio からの出力機能の開発
- 埋め込み先プログラムから Data Tables や閲覧者の入力を受け取り，Rendering 処理へ描画パラメータを引き渡すためのインタフェース関数の設計

5.2.2 実行環境の選定

Visual Mappings および View Transformations 処理のための実行環境に求められる要件は以下の通りである．

R1. 埋め込み先のプログラムから容易に処理を呼び出せる

R2. 埋め込み先で実装された描画関数等の登録が容易

R3. 汎用的な演算が可能

R1 と R2 は，埋め込みを容易にするために必要な要件である．一方，R3 は，ノードの雛形の拡張性を考慮した要件である．本研究では，目標 2 で設定したように，ノードの雛形をプログラミング言語によって拡張できるようにすることを目標としている．そのため，Iv Studio の利用者がノードの雛形を拡張した場合，実行環境ではその演算処理を実行できなければならない．

この要件を全て満足する方法について，以下のような候補を検討した．

[仮想マシンを利用]

JavaVM や.NET Frameworks（共通言語基盤の実装）で動作する実行ランタイムを用意し，埋め込み先のプログラムとソケット通信やプロセス間通信を使い，データや描画パラメータの受け渡しをする方法が考えられる．Iv Studio 上で実装した可視化処理をその実行ランタイムとリンクできる動的リンクライブラリとして出力することで，可視化処理をプログラムに埋め込んだように動作させることが可能である．しかし，埋め込み先のプログラムに，実行ランタイムのプロセスを起動する処理や，通信処理を実装する必要があるため，R1 や R2 に対して不十分である．また，JavaScript で書かれたプログラム（Web ページ）との通信も実装が難しい．

[C 言語のソースコードを出力]

Iv Studio から C 言語のソースコードを出力し，プログラムに埋め込むことができるようにする方法が考えられる．先の方法のプロセス起動や通信の処理の実装の手間を解消した方法である．C 言語で書かれているプログラムからはそのまま利用することができ，Java や C#からは，Java Native Interface[JNI] 等のネイティブコードとの連携インタフェースを用いて利用することができる．JavaScript については，Emscripten[ems] を用いて利用が可能である．しかし，この方法も，Java や C#等からの呼び出しや関数の登録の容易さに関して懸念が残るため，R1 や R2 に対して不十分である．

[組み込み用スクリプト言語を利用]

ソフトウェアライブラリとして言語の実行環境が提供されている組み込み用スクリプト言語を利用する方法が考えられる．Iv Stuido からスクリプト言語のソースコードを出力し，組

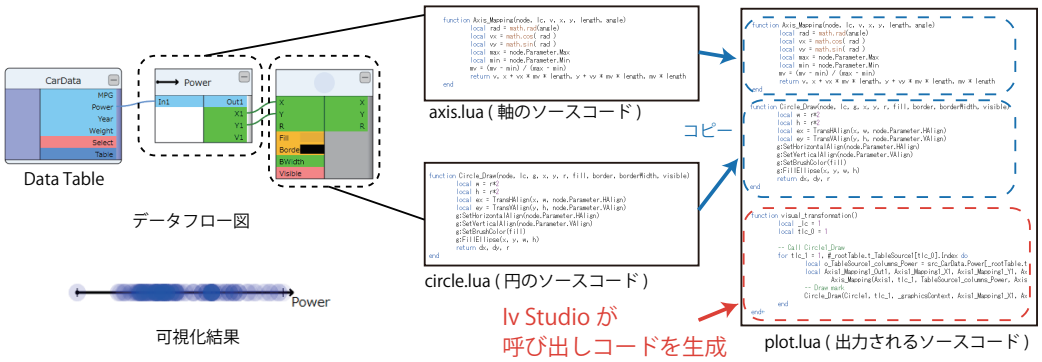


図 5.3: ソースコードの生成手順

み込み先のプログラム上でそのソースコードを実行することで，様々なプログラムから利用することが可能である．組み込み用スクリプト言語は，組み込み先の言語との連携を前提として設計されているため，C 言語のソースコードを出力する方法と比べ，呼び出しや関数の登録が容易である．この方法は，要件全てを満足する．

以上の検討を経て，本研究では，実行環境に組み込み用スクリプト言語を採用した．組み込み用スクリプト言語には，Lua[luac]，Squirrel[Squ]，AngelScript[Ang]，Xtal[Xta] など様々なものがあるが，想定する C，Java，.NET，JavaScript 用の実装があるうえ，開発資料や利用実績の多い [浜中 08]Lua を採用した．それに伴い，Iv Studio のノードの雛形の記述言語にも Lua を採用し，Lua を用いてノードの雛形を拡張できるようにした（詳細は 5.2.5 にて述べる）．

5.2.3 Iv Studio からの出力機能とサポートライブラリ

可視化処理を Lua の実行環境（LuaVM）上で実行できるよう，Iv Studio に Lua のソースコード出力機能を実装した．ノードの処理は全て Lua で記述し，Iv Stuido で記述したデータフロー図に従って，ノードの処理の関数を呼び出すルーチンのソースコードを生成できるようにした．そして，開発者が容易に利用できるよう，生成したソースコードとノードの処理関数を 1 つのファイルにまとめて出力できるようにした．図 5.3 にソースコードの生成手順を示す．

Iv Studio から出力されるソースコードは Visual Mappings 処理のみである．それ以外の View Transformations 処理や，マークの選択状態の逆伝搬処理に関しては，Lua で書かれたサポートライブラリを提供した．

Iv Studio の利用者は，埋め込み先のプログラムに Lua のライブラリをリンクし，サポートライブラリと Iv Stuido から出力されたソースコードを読み込んで実行することで，可視化処理を埋め込み先のプログラム上で実行できる．くわえて，この実行環境は，埋め込み先のプログラムを実行した状態でも，可視化処理を更新することが可能である．このことは，可視化手法の修正が繰り返されることが想定される可視化プログラムの開発において，大きな利

表 5.1: データテーブル用関数

関数名	仕様
obj OpenTable(filename, format)	ファイル名等と形式（タブ区切り，カンマ区切り）を受け取り，参照用オブジェクトを返す
string ReadRecord(obj)	1 レコード分のデータを，タブ区切り，または，カンマ区切り形式の文字列で返す
void CloseTable(obj)	テーブルを閉じる

点である．

5.2.4 インタフェース関数の設計

可視化の実行には，LuaVM と埋め込み先のプログラムの間で，Data Tables の受け取りと Rendering 処理への描画パラメータの引き渡しが必要である．埋め込みを容易にするためには，これらを行うためのインタフェース関数の実装コストを下げることが重要である．そのため，最低限実装が必要な関数の数を抑え，かつ，その関数の実装も簡潔に行えるようなインタフェース関数を設計した．

数を抑えるため，命令は低レベルなものにし，Lua のサポートライブラリ側で多くの処理を行うようにした．また，関数の実装を容易にするため，関数の機能を大半の言語や描画 API に備わっている命令のみで，そのほとんどを実装できるようにした．さらに，LuaVM との配列の受け渡しは記述が煩雑になることから，引数や返り値に配列を利用することを避けた．

まず，表 5.1 に，データテーブル用関数の仕様を示す．実際には，ファイル読み込みや HTTP 通信用に別の関数が用意されており，関数名も ivs_OpenFile，ivs_ReadLine，ivs_Close のように異なるが，説明の都合上，機能を表すような関数名を示している．開発者は，これらの関数を埋め込み先のプログラミング環境を用いて実装し，LuaVM に登録することで，データテーブルの受け渡しが可能になる．関数の実装は簡潔に行える．例えば，C 言語では，ivs_OpenFile で fopen を呼び出し，ivs_ReadLine で fread，ivs_Close で fclose を呼び出せば，タブ区切りやカンマ区切り形式のファイルの読み込みを実装できる．必ずしもファイルの入出力を実装する必要はなく，例えば，ivs_OpenFile で SQL のクエリを発行し，ivs_ReadLine で結果を 1 レコードずつ返すようにすることで，SQL データベースとの連携も実装できる．

次に，表 5.2 に，実装および LuaVM への登録が最低限必要な描画用関数の仕様を示す．表の上 4 つは，色やフォントサイズの指定関数，下 5 つが描画を行う関数である．上 4 つの関数は，下 5 つの関数に統合することも考えられた．例えば，線を描画する関数は DrawPath のみのため，この関数の引数に色と線の太さを渡す仕様にすることで，SetPenColor と SetPenWidth は不要となる．しかし，次に述べるオプショナルな描画用関数の実装コストを下げるために，指定関数と描画を行う関数を別に分けた．

描画処理の実行パフォーマンスの向上のために，開発者による実装および登録がオプショ

表 5.2: 最低限実装が必要な描画用関数

関数名	仕様
void SetPenColor(r, g, b, a)	描画する線の色を設定する
void SetPenWidth(w)	描画する線の太さを設定する
void SetBrushColor(r, g, b, a)	描画する塗り色を設定する
void SetFont(size)	描画するフォントサイズを設定する
void BeginPath(x, y)	描画パスを開始する
void MoveTo(x, y)	描画パスに点を追加する
void DrawPath(isClose)	設定した描画パスに従い線を描く
void FillPath()	設定した描画パスに従い多角形を塗りつぶす
void DrawText(x, y, text, angle, rx, ry)	指定した位置に，文字列を描画する

表 5.3: オプショナルな描画用関数

関数名	仕様
void DrawLine(x1, y1, x2, y2)	直線を描画
void DrawRect(x, y, w, h)	矩形の枠線を描画
void FillRect(x, y, w, h)	塗りつぶした矩形の描画
void DrawEllipse(x, y, w, h)	楕円の枠線を描画
void FillEllipse(x, y, w, h)	塗りつぶした楕円の描画
void DrawWedge(x, y, iRad, oRad, startAngle, sweepAngle)	楔形の枠線を描画
void FillWedge(x, y, iRad, oRad, startAngle, sweepAngle)	塗りつぶした楔形の描画

ナルな描画用関数も用意した．表 5.3 に，オプショナルな描画用関数を示す．表 5.2 の関数と比べて抽象度が高いが，大半の描画 API が持つ命令のみで，簡潔に実装できる．開発者によってこれらの関数が登録されていない場合，Lua のサポートライブラリによって多角形の描画命令に分解されて実行される．しかし，描画 API に用意されている関数を直接用いる場合と比べて，実行パフォーマンスの低下が予想される．そこで，開発者が，プログラムに埋め込む可視化処理に応じて，実装コストと実行パフォーマンスのどちらを重視するかを選択できるようにした．

最後に，表 5.4 に，インタラクション手法の動作のために呼び出しが必要な入力の受け付け関数を示す．これらの関数は全て，ポインタの座標やマウスホイールの動作量，操作したマウスボタンの情報を受け取り，実行環境が何らかの処理を行った場合は True を返す．開発者は，入力イベントの発生時にこれらの関数を呼び出すだけで，インタラクション手法を動作させることが可能である．マウスに限らず，タッチパネルを備えたシステムを用いた場合も，適切なタイミングで呼び出すことで利用可能である．

表 5.4: 入力の受け付け関数

関数名	呼び出しタイミング
boolean OnMouseDown(x, y, wheel, button)	マウスボタンのダウン時に呼び出し
boolean OnMouseMove(x, y, wheel, button)	ポインタの移動時に呼び出し
boolean OnMouseUp(x, y, wheel, button)	マウスボタンのアップ時に呼び出し
boolean OnMouseWheel(x, y, wheel, button)	マウスホイールの操作時に呼び出し

5.2.5 ノードの雛形の拡張方法

本項では，Lua を用いた Iv Studio のノードの雛形の拡張方法について説明する．ここでは基本的な定義方法と例を紹介し，Iv Studio の利用者がノードの雛形を柔軟かつ容易に拡張できることを示す．

ノードの雛型のための Lua スクリプトファイルには，ノードの雛型の定義関数，ノードの生成関数，ノードの関数の更新関数，ノードの処理関数の 4 つの関数を最低限記述する必要がある．以下に，ノードの雛型のための最低限のテンプレートを示す．Lua の言語仕様について簡単に説明すると，関数の定義には function キーワードを用い，−−[[ ]] はソースコードのコメントを表す．

```
function define_node()
    local nodeType = NodeType:new("MyNode")
    return nodeType
end

function create_node(nodeType, name)
    local node = Node:new(nodeType, name)
    return node
end

function update_functions(node)
    local func = NodeFunction:new("Func")
    node.TransFunctions = { func }
    return node
end

--[[FUNCTIONS]]
function MyNode_Func(node, lc)
end
```

define_node がノードの雛型の定義関数，create_node がノードの生成関数，update_functions がノードの関数の更新関数である．これらの関数名は固定である．define_node 関数では，NodeType:new 関数を呼び出してノードの雛型のオブジェクトを生成し，それを返す．その際，ノードの雛型の名前を指定する（例では「MyNode」としている）．create_node 関数では，引数に生成するノードの雛型のオブジェクトと，ノードを識別するための名前が渡される．これらの引数を Node:new 関数に渡し，ノードのオブジェクトを生成して，それを返す．update_functions 関数では，ノードの関数やピンの情報を設定する．この関数は，Iv Studio 上でノードが生成された際やピンが接続された際，ノードのパラメータが編集された際に，そのノードのオブジェクトが引数に渡されて呼び出される．この例では，NodeFunction:new 関数を使って Func という関数を生成し，ノードに設定している．詳細は後述する．

「−−[[FUNCTIONS]]」コメント以降は，Iv Studio から出力される Lua のソースコードにコピーされる．ここに，ノードの処理関数を記述する．処理関数の名前には，define_node と update_functions で設定したノードの雛形名と関数名を_（アンダーバー）でつないだ，「[ノードの雛形名]_[関数名]」が用いられる．そのため，ノードの雛形名が重複しない限り，処理関数名が重複することはない．

次に，楕円のマークを描くためのノードの雛型の例を示し，具体的な記述方法を説明する．

```
function define_node()
    local nodeType = NodeType:new("Ellipse")

    nodeType.Editor.BorderWidth = EditorInfo:new()
    nodeType.Editor.BorderWidth.Type = "Float"
    nodeType.Editor.BorderWidth.MinValue = 0

    nodeType.Editor.Fill = EditorInfo:new()
    nodeType.Editor.Fill.Label = "Fill Color"
    nodeType.Editor.Fill.Type = "Color"

    return nodeType
end

function create_node(nodeType, name)
    local node = Node:new(nodeType, name)

    node.Parameter.BorderWidth = 1
    node.Parameter.Fill = Color:new(0, 0, 1, 1)

    return node
```

```
end

function update_functions(node)
    local func = NodeFunction:new("Draw")
    node.TransFunctions = { func }

    func.Category = "Mark"

    table.insert(func.Inputs, Pin:new("X", "Spatial"))
    table.insert(func.Inputs, Pin:new("Y", "Spatial"))
    table.insert(func.Inputs, Pin:new("W", "Spatial"))
    table.insert(func.Inputs, Pin:new("H", "Spatial"))

    table.insert(func.Outputs, Pin:new("X", "Spatial"))
    table.insert(func.Outputs, Pin:new("Y", "Spatial"))
    table.insert(func.Outputs, Pin:new("W", "Spatial"))
    table.insert(func.Outputs, Pin:new("H", "Spatial"))

    return node
end

--[[FUNCTIONS]]
function Ellipse_Draw(node, lc, g, x, y, w, h)

    g:SetBrushColor(node.Parameter.Fill)
    g:FillEllipse(x, y, w, h)

    if node.Parameter.BorderWidth > 0 then
        g:SetPenColor( Color:new(0, 0, 0, 1) )
        g:SetPenWidth(node.Parameter.BorderWidth)
        g:DrawEllipse(x, y, w, h)
    end

    return x, y, w, h
end
```

---

define_node 関数には，主に Iv Studio 上でのノードのパラメータの設定方法を記述する．Lua にはテーブル（連想配列）の言語仕様があり，ノードの雛型のオブジェクトはテーブルであ

る．NodeType:new 関数で生成されたテーブルには，メンバとして Editor というテーブルが追加されており，その Editor のメンバに EditorInfo:new で生成したオブジェクト（テーブル）を追加することで，Iv Studio のパラメータパネルからノードのパラメータを設定できるようになる．Editor に追加するメンバ名は，後述する create_node 関数で設定するノードのパラメータ名と一致している必要がある．

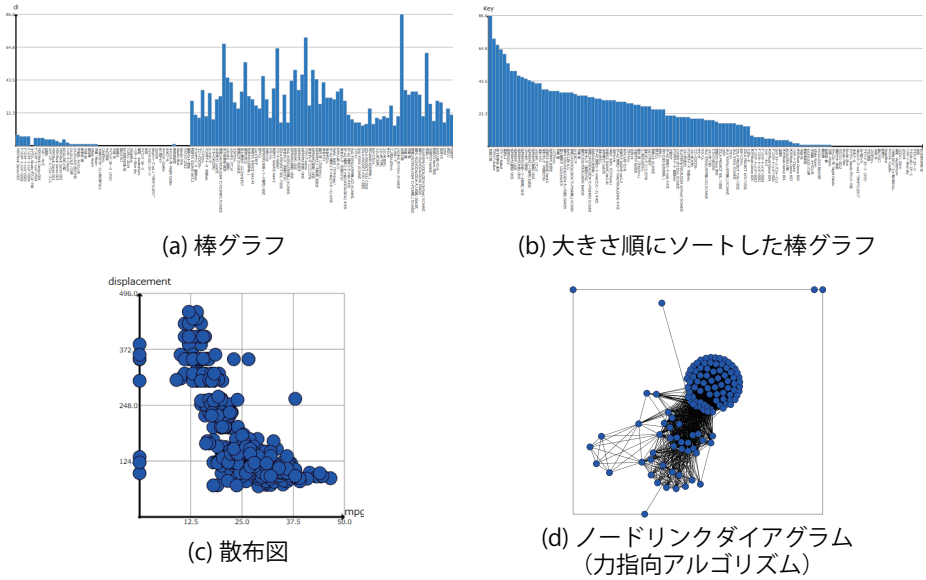
create_node 関数には，ノードのパラメータとそのデフォルト値を，ノードのオブジェクトの Parameter のメンバに追加するように記述する．この例では，楕円の枠線の太さ（BorderWidth）と塗り色（Fill）のパラメータを用意している．Parameter には，create_node 関数や処理関数で使用するための値を設定するのに対して，Editor には Iv Studio からその値をどのように編集できるようにするかを設定する．例えば，「nodeType.Editor.BorderWidth.Type = ”Float”」では，BorderWidth パラメータを実数設定用のユーザインタフェースで設定できるようにすることを指定おり，「nodeType.Editor.BorderWidth.MinValue = 0」にて最小値を指定している．「nodeType.Editor.Fill.Type = ”Color”」では，Fill パラメータを色指定用のユーザインタフェースで設定できるように指定している．このように，Iv Studio 上でのパラメータの編集方法に関する細かな指定も行える．

update_functions 関数には，関数のオブジェクトの生成や，そのオブジェクトに対する関数の種類や入力ピンと出力ピンの設定を記述する．例では，まず，「func.Category = ”Mark”」という記述で func が描画関数であることを設定している．関数の種類が Mark に指定されると，処理関数の引数に描画用のオブジェクトが渡されるようになる．次に，関数のオブジェクトの Inputs メンバに，入力ピンを追加する．Pin:new 関数を用いてピン用のオブジェクトを生成して，それを追加する．Pin:new 関数には，ピンの名前とデータ型を指定する．ここでは，座標値を表す “Spatial” を指定している．入力ピンと同様に Outputs メンバに，出力ピンを追加する．

処理関数には，1 レコード分のデータの処理を記述する．これは，D3[BOH11] のデータドリブンと同様のパラダイムである．処理関数 Ellipse_Draw には，第 1 引数にノードのオブジェクトが渡され，このオブジェクトからノードのパラメータを参照することができる．第 2 引数には描画対象のレコードのインデックス番号，第 3 引数には描画用のオブジェクトが渡される．描画用オブジェクトには，一般的な描画 API 相当の命令が用意されており，それを用いて描画処理を記述する．第 4 引数以降には入力ピンで指定した順に値が渡される．この例では，X ピン，Y ピン，W ピン，H ピンの順に，Iv Studio 上で記述された入力が渡され，その値を用いて楕円を描く処理を記述している．戻り値は，出力ピンに設定した通りの順序で値を返す．Lua には複数の値を戻り値として返す仕様があるため，それを利用している．

5.3 実行パフォーマンスの評価

設計した実行環境では，可視化処理の実行を LuaVM 上で行う．Lua はスクリプト言語の中では比較的動作が高速 [浜中 08] なものの，ネイティブコードの実行速度と比べると劣ると考えられる．設計した実行環境が，可視化処理を扱うのに十分なパフォーマンスを有している



(a) 棒グラフ　(b) 大きさ順にソートした棒グラフ

(c) 散布図　(d) ノードリンクダイアグラム（力指向アルゴリズム）

図 5.4: 実行パフォーマンスの評価に用いた可視化手法

かどうかを確認するために，実行パフォーマンスを評価する実験を行った．

まず，図 5.4 に示す可視化のパフォーマンス評価を行った．(a) は棒グラフ，(b) は値の大きさ順にソートした棒グラフ，(c) は散布図，(d) は力指向のレイアウトアルゴリズム [Ead84] を用いた一般グラフのノードリンクダイアグラムである．用いたデータの規模を，表 5.5 に示す．実行環境には，C（C++）と JavaScript の実装を用意した．C（C++）の Lua ライブラリには，Lua バイトコードを実行時コンパイルして動作させる Lua のライブラリ，LuaJIT[luab] を用いた．LuaJIT のコンパイルには，Visual Studio 2015[vs2] を用いた．JavaScript の Lua ライブラリには，C 言語向けの Lua ライブラリを Emscripten を用いて JavaScript 用のライブラリにコンパイルした lua.vm.js[luaa] を用いた．また，JavaScript の実行には，Electron 1.4.1[ele] を用いた．どちらも，OS が Windows 10，CPU が Intel Core i7-6700K の環境上で動作させ，Iv Studio から出力した Visual Mappings 処理にかかる時間を測定した．

表 5.5 に実験結果を示す．時間の単位はミリ秒で，100 回の試行の平均時間を示している．ただし，5.4(d) のみ，レイアウト完了までの時間ではなく，レイアウトアルゴリズムの 1 回のイテレーションにかかる時間を示している．

インタラクティブ手法の動作を想定した場合，30 フレームレート以上で動作すれば十分なパフォーマンスといえる．ただし，今回の実験では，描画処理の時間を含めていない．筆者のこれまでの可視化システムの開発の経験上，描画処理は Visual Mappings 処理よりも時間がかかることが多い．その分を考慮し，この実験では，10 ミリ秒以下であれば十分なパフォーマンスと判断した．

実験の結果，C（C++）環境下では，3 ミリ秒未満で動作し，十分なパフォーマンスであることが分かった．データの規模が大きい (d) においても，高速に処理できた．これは，アル

表 5.5: 実行パフォーマンスの評価結果（単位:ミリ秒）

手法	データの規模	C（LuaJIT）	JavaScript（lua.vm.js）
図 5.4(a)	123 レコード	0.87	5.19
図 5.4(b)	123 レコード	0.83	5.52
図 5.4(c)	406 レコード	1.43	8.53
図 5.4(d)	123 ノード 4094 エッジ	2.96	70.13

ゴリズムの単純な繰り返し処理が，LuaJIT の実行時コンパイルの影響で最適化された効果によるものと考えられる．JavaScript 環境下においても，(a) から (c) までは，十分に実用的なパフォーマンスで動作した．一方，(d) の処理には，時間がかかった．今後は，Iv Studio が出力する Lua のソースコードの最適化が必要になると考えられる．ただし，JavaScript を用いて今回の可視化処理を実装し，その実行時間を計った場合に，どの程度の処理時間になるかは調査できていないため，JavaScript による実装と，LuaVM 上での処理時間の比較が必要である．

次に，データのレコード数の観点からパフォーマンス評価を行った．0 から 100 のランダムな実数値を 2 つ持つ，2 次元のデータを，散布図によって可視化する処理を作成し，その処理時間を計測した．実験環境には，先の実験と同様に LuaJIT と lua.vm.js を用いた．データのレコード数を 500 から 50000 まで，500 ずつ増やしていき，各データレコード数ごとに 100 回試行し，その実行時間を記録した．

実験結果を，図 5.5 に示す．この図では，X 軸がデータのレコード数，Y 軸が 100 回の試行の平均実行時間を示しており，青が LuaJIT，緑が lua.vm.js の結果を示す．(a) と (b) は，同じデータを表す図であるが，分かりやすさのためにスケールを変えたものを示した．lua.vm.js では，23000 レコードの段階でメモリ不足となり，それ以降は実行できなかった．そのため，記録が 22500 レコードまでとなっている．このメモリ不足とは，lua.vm.js を Emscripten でコンパイルした際に設定されたメモリ領域を超えたことを意味しており，設定を変更して再コンパイルすることで，より大きなデータを取り扱うことができるようになると考えられる．

まず，LuaJIT の結果について述べる．インタラクティブ手法の動作を想定し，10 ミリ秒程度で処理できたレコード数を述べると，9000 レコードを 10.11 ミリ秒で処理することができた．静的な可視化を取り扱うことを想定した場合には，50000 レコードを 53.50 ミリ秒で処理できることから，比較的大規模なデータに対しても現実的な速度で利用可能と言える．近年の情報可視化では，生データの段階で見ると，数 100 万レコードを超えるデータや，数十次元のデータを取り扱うこともあるが，このようなデータの可視化では統計処理やフィルタリング，次元圧縮，クラスタリングなどの技術によってデータが削減されることが多い．このような削減処理は，Iv Studio で主に取り扱う Visual Mappings 以前の処理であり，Visual Mappings 処理への入力段階ではレコード数や次元数は削減されていることが想定される．そのため，インタラクティブな手法を想定して 9000 レコード，静的な可視化を想定して 50000 レコード程度の取り扱いが可能であれば，十分に実用的と言える．

次に，lua.vm.js の結果について述べる．10 ミリ秒程度で処理できたレコード数に関しては，

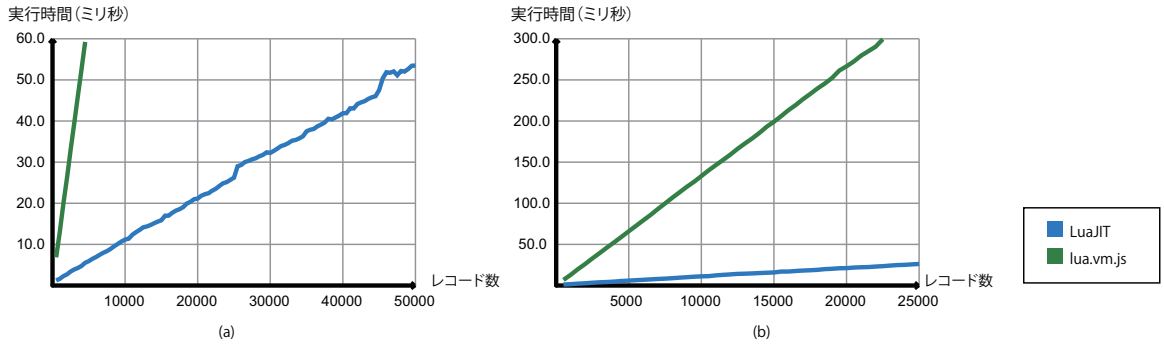


図 5.5: レコード数と実行時間の関係の評価結果

1000 レコードのデータに 12.99 ミリ秒の処理時間がかかった．インタラクティブ手法を Web ページ上で取り扱う場合には，この程度のデータ量までしか取り扱いが難しいと言える．ただし，Web サイト上でのプレゼンテーション用途として可視化を利用するといった数十程度の少量のデータを取り扱うような利用方法としては，十分な処理速度と言える．また，静的な可視化を想定するならば，JavaScript を用いてクライアントサイドで処理するのではなく，サーバサイドで SVG を生成して返すといったシステムの実装方法が考えられる．そのようなシステムの実装にも，開発した実行環境を利用することができるため，必ずしも数万レコードを JavaScript で処理する必要はないと考えられる．

5.4 利用事例

Iv Studio と設計した埋め込み可能な実行環境を，実システムに利用した例を 2 つ紹介する．いずれのシステムも，筆者が開発したものであるが，本研究のために開発したシステムではなく，別の目的で開発されていたものに対して Iv Studio を利用した点を強調しておく．つまり，既存のシステムに可視化の機能を追加するという利用方法の例と言える．

5.4.1 Web ページ

一つ目の例は，筆者の運営する，デジタルコンテンツのダウンロードサイト [cha] の管理ページへの利用である．筆者は，兼ねてより，この管理ページにダウンロードの時間帯や数などを一望できるような可視化の機能を加えたかった．そこで，Iv Stuido を用い，状況を一望することのできる可視化の機能を追加した（図 5.6）．

まず，サーバサイドのプログラムに，SQL データベースから一定期間分のダウンロード情報の一覧と，コンテンツ情報の一覧を CSV 形式で出力する機能を追加した．そして，その CSV 形式のデータを可視化する部分を Iv Studio で作成した．制作した可視化の機能では，ChronoView を左側に配置してコンテンツのダウンロード時間を表示し，右側にはコンテンツ名とダウンロード数を示す横棒を表形式で並べて表示するようにした．さらに，マークを選

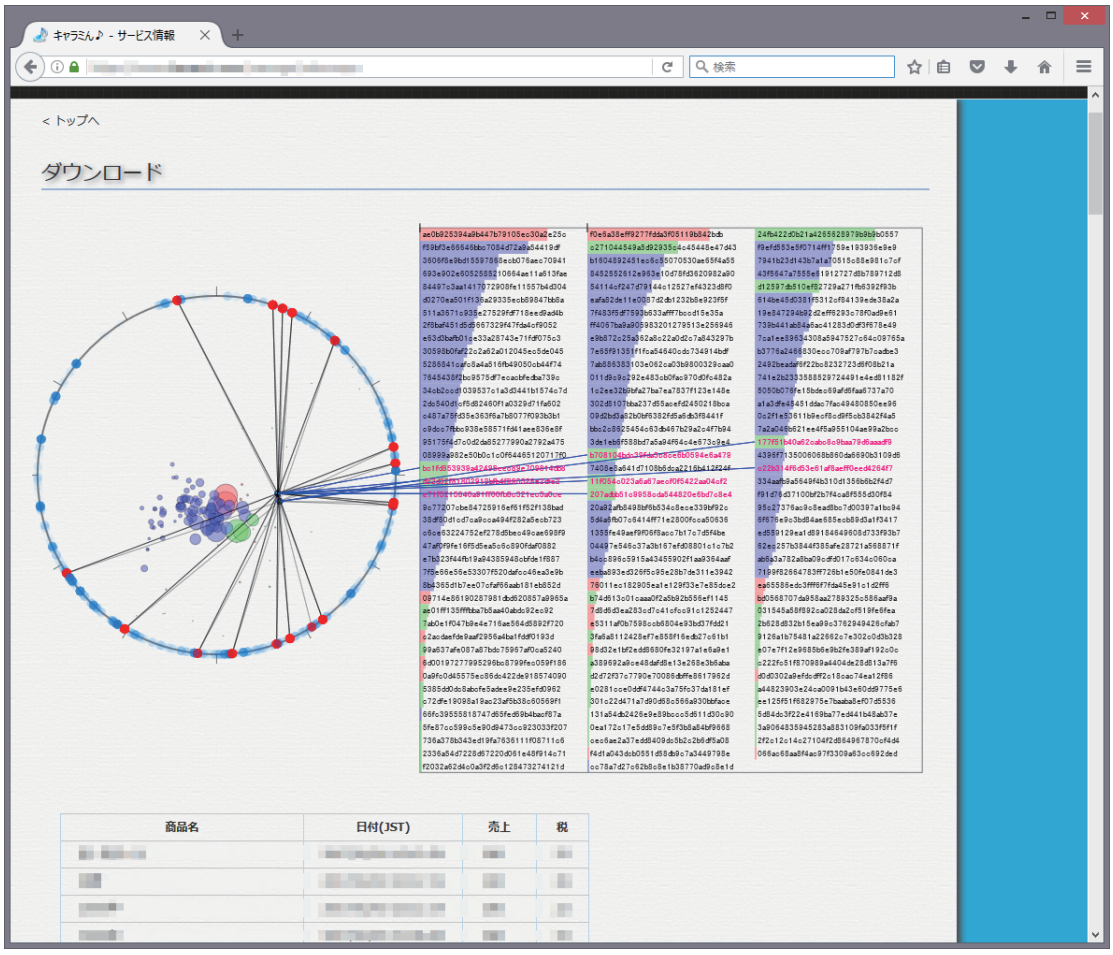


図 5.6: 可視化の機能を追加した Web ページ

択すると，左の ChronoView と右の表との間を青い線でつなぐようにし，ChronoView の円周上の青い円とも灰色の線でつなぐようにした．この機能の制作において，使用する可視化手法や配色などの様々な試行錯誤を行った．記述したデータフロー図を図 5.7 に示す．記述を完了後，Iv Studio から Lua のソースコードを出力し，管理サイトのページ上で JavaScript を用いて動作させた．

筆者は，JavaScript や Canvas に関して知識はあったものの，実際にそれらを使って開発した経験はなかった．そのため，制作したような複雑な可視化手法をサイト上に実装するには，Iv Studio を使うよりも時間がかかったと考えられる．

JavaScript に不慣れなことや，lua.vm.js の資料が少なかったことから，LuaVM への関数の登録や関数の呼び出しが行えるようになるまでに 3 から 4 時間程度の時間を要した．なお，この後，Iv Studio に，可視化処理の Lua ソースコードと共に，この際に開発した JavaScript 用のサポートライブラリと HTML ファイルを出力する機能を追加し，可視化処理を Web ページ上でより容易に利用できるようにした．

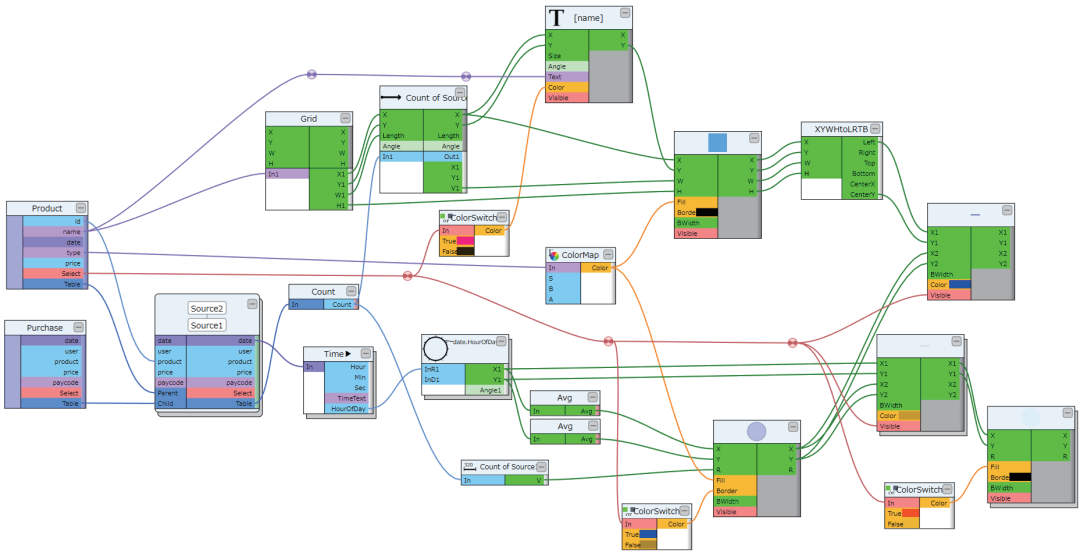


図 5.7: Web ページに追加した可視化手法を記述したデータフロー図

5.4.2 デバイス開発のテストプログラム

もう一つの例は，筆者の開発している，ポーズキャプチャデバイスのテストプログラムへの利用である．開発しているデバイスは，複数の 3 軸加速度センサと 3 軸地磁気センサを備えた人形型のデバイスである．テストプログラムは，デバイスからセンサ情報を読み取り，ポーズを計算し，3D モデルに反映させて表示するプログラムである．開発中，地磁気センサからのポーズ推定にズレがあり，その原因解明のために，センサから取得したデータを可視化する機能をテストプログラムに追加した（図 5.8）．

このテストプログラムは，C++で書かれており，独自の描画システムを持ったツールキット上に構築されていた．このツールキット用の描画関数を実装することで，可視化機能を 30 分程度で追加することができた．

制作した可視化の機能について説明する．X-Y, Y-Z, X-Z それぞれの面から見た地磁気センサの値をプロットする図を置き，右下に鉛直方向から見たセンサ値を表す図を置いた．プロット図には，加速度センサから取得した鉛直方向を橙色の線で表現するようにした．ここまで制作し，Iv Studio から出力して，テストプログラムに埋め込んだ．テストプログラムで表示する際には，プログラムで計算したポーズと見比べられるよう，作成した表現をポーズ画面の上に半透明で重畳表示するようにした．このような半透明の重畳表示は，埋め込み先のプログラムの使用している描画 API と同じものを用いて可視化結果を描画するようにしたことで可能になったものである．

この制作では，テストプログラム上で可視化結果を確認しながら，半透明の具合や色合いの調整を頻繁に行った．Iv Studio 上でパラメータを調整し，テストプログラムを実行したまま Lua スクリプトのリロードを行うことで，効率的に調整を行えた．

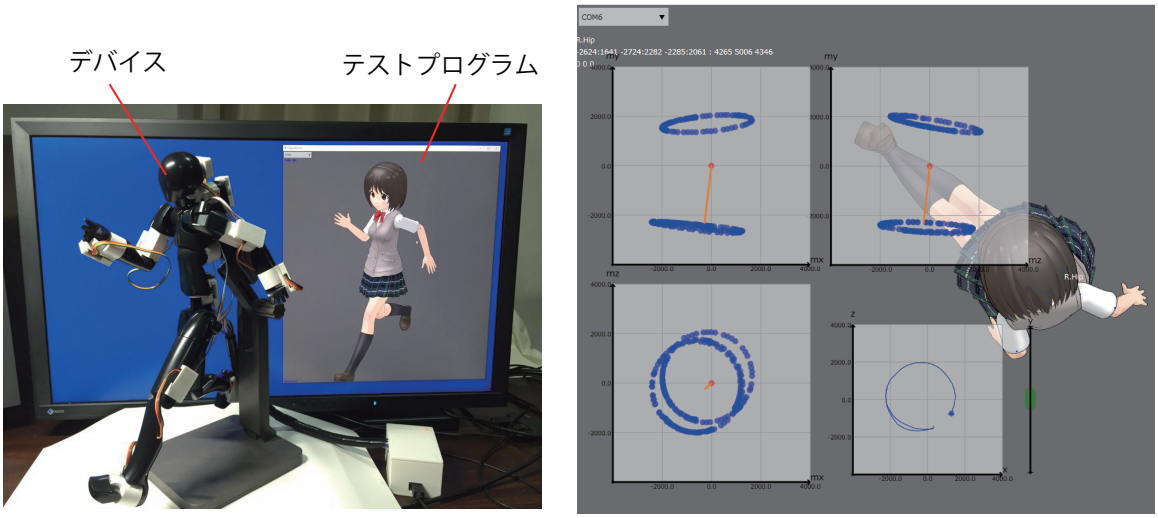


図 5.8: 開発中のデバイスと地磁気センサの情報を可視化する機能を追加したテストプログラム

第6章　ケーススタディ: 時間軸を用いた量的データの可視化手法

本章では，時間軸を用いた量的データの可視化手法に関する研究と，その研究に対する Iv Studio の利用について述べる．まず，研究の概要と提案する可視化手法について述べる．次に，提案手法からの情報の読み取りに関する実験と，量的データの表現精度に関する実験について述べる．最後に，時間軸を用いた量的データの可視化手法に関する研究における Iv Studio の利用方法と，それに関する考察を述べる．

6.1　研究の概要

可視化手法の設計の難しさの要因一つに，値を表現することのできる視覚的属性の種類の少なさがある．種類の少なさから，手法の設計者には，位置や色の使い方の工夫が求められる．その難しさの軽減のため，この研究では値の表現に利用可能な視覚的属性の種類を増やすことを目的とする．

視覚的属性の種類を増やす手段として，時間軸の利用，すなわち，アニメーションは有効な手段の一つである．マークの大きさや角度変化の頻度によって量的データを表現できると考えられる．もし，アニメーションによって量を表現することができれば，視覚的属性の種類を増やすことができる．しかしながら，頻度を用いた量的データの表現は実際に見られるものの，その利用方法や表現の精度についてはまだ調査がなされていない．

この研究では，繰り返し再生される短いアニメーションをループアニメーションと呼び，その利用方法や量的データの表現精度に関する調査を行った [IM16]．まず，量的データの表現のためのループアニメーションのバリエーションを従来の静的な視覚変数に基づいて作成した．次に，ループアニメーションからの情報の読み取りや量的データの表現精度に関する調査のために，2 つの実験を行った．

6.2　ループアニメーションを用いた量的データの可視化手法

ループアニメーションを用いた量的データの可視化手法について説明する．図 6.1(a) は，マークが回転するループアニメーションを用いた例である．この例では，量的な値によって回転の頻度（速度）を変化させることで，量を表している．図 6.1(b) は，マークを上下に振動させることで量を表現する例である．この例では，量的な値によって振動の頻度と振幅の

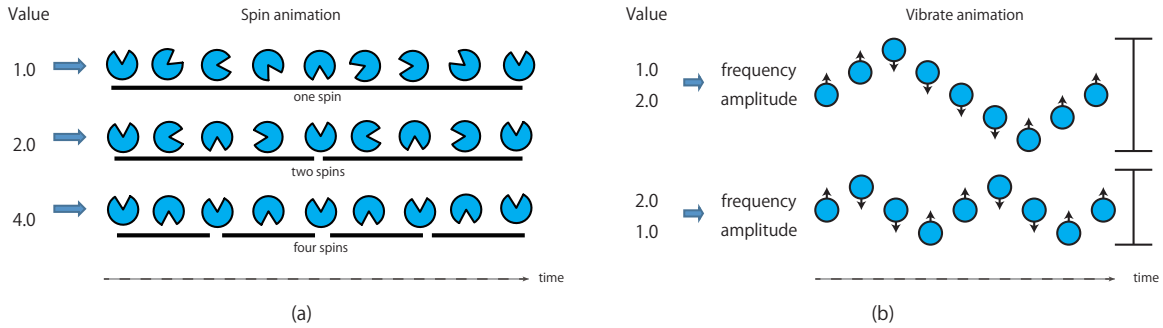


図 6.1: ループアニメーションの例

両方を変化させることで，2 つの量を表している．このように，ループアニメーションの頻度や振幅によって，量的データを表現することができると考えられる．

筆者は，ループアニメーションが既存の可視化手法と共に用いられることを想定している．可視化手法は，Bertin の視覚変数 [Ber84] を参考に設計されることが多い．そこで，Bertin の視覚変数に基づいた量的データを提示するためのループアニメーションのバリエーションを提案する．

表 6.1: ループアニメーションのバリエーション

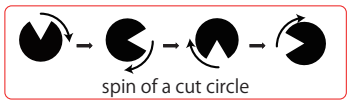
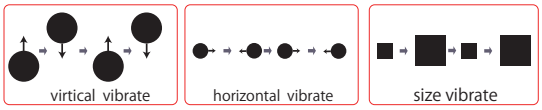
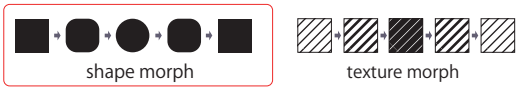
バリエーション	視覚変数	例
Spin	Orientation	
Blink	Value, Color	
Vibrate	Position, Size	
Morph	Shape, Texture	

表 6.1 の各行に，ループアニメーションのバリエーションを示す.「視覚変数」の列には，各バリエーションのループアニメーションに用いる Bertin の視覚変数を示し,「例」の列には，

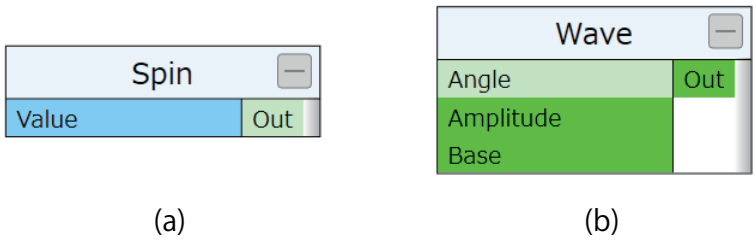


図 6.2: Spin ノードと Wave ノード

ループアニメーションの具体例を示す．Spin は時間に応じてマークの向き（Orientation）を変化させるアニメーションである．同様に，Blink はマークの明るさ（Value）や色相（Color）を，Vibrate は位置（Position）や大きさ（Size）を変化させるアニメーションである．Morph は，マークの形（Shape）やきめ（Texture）のモーフィングレベルを時間に応じて変化させるアニメーションである．

ループアニメーションを用いた量的表現には，以下のような利点があると考えられる．

1. 従来の視覚変数を 1 つ用いて，複数の変量を表すことができる．例えば，位置の Vibrate アニメーションでは，振動の起点となる位置と，振動の頻度，振幅によって 3 変量を表すことができる．

2. 頻度や速度に関するデータの意味を閲覧者に対して直感的に表すことができる．例えば，Spin の速さは速度という意味を直感的に表すことができる．

6.3 ループアニメーション表現のためのノードの雛型

後述するループアニメーションの評価実験のために，Iv Studio にループアニメーションを記述するためのノードの雛型を作成した．提案したバリエーションを基準に 4 種類の雛型を作成することも検討したが，別の方針として，必要な機能を基準に考えた場合には，値を回転速度に変換する機能と，何らかの波形に従って角度を数値や座標値に変換する機能の 2 つを用意するだけで，すべてのバリエーションを記述することができるため，必要な機能を基準として，2 つのノードの雛型を作成した．

まず，値を回転速度に変換するノードの雛型 Spin を作成した（図 6.2(a)）．このノードは，数値を入力として，時間に応じて変化する角度を出力する．出力される角度の回転速度は入力された数値によって変化する．回転速度の計算には，式 6.1 を用いた．この式では，回転速度の係数を s として，ある数値 v を入力としたときの，経過時間 T における角度 $A(v,T)$ が求められる．この式を用いることで，例えば，T に秒単位の経過時間を入力するようにしたとき，svHz の速度で回転する角度が得られる．

$$A(v,T) = 2\pi T(sv) \tag{6.1}$$

この式を Spin の処理関数に実装した．以下に，そのコードを示す．

```
function Spin_Spin(node, lc, value)
    return (AnimationTime * 360 ) * (value * node.Parameter.Speed)
end
```

コード中の AnimationTime は，可視化処理を開始してからの経過時間を取得できるグローバル変数で，node.Parameter の Speed は回転速度の係数 s の変数である．Speed は，Iv Studio のノードパネルから設定できる．そのため，開発者は，Iv Studio 上で，回転速度をインタラクティブに調整できる．なお，Iv Studio では，角度を度分単位で取り扱っているため，2π の代わりに 360 を掛けている．

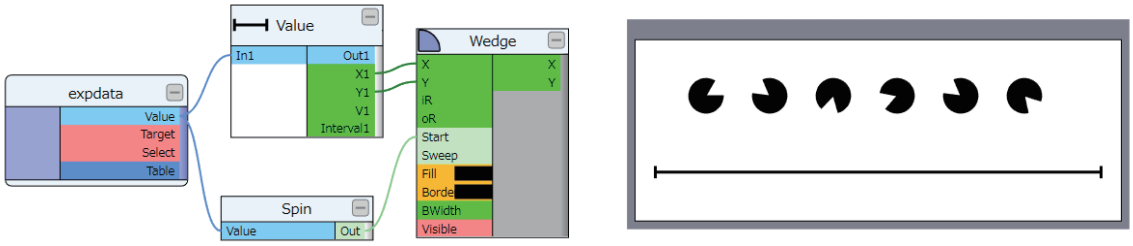
次に，何らかの波形に従って角度を数値や座標値に変換するノードの雛型 Wave を作成した（図 6.2(b)）．このノードは，変換元となる角度，変換に必要な波の中心点，波の振幅の，3 種類の値を受け取り，数値型もしくは座標型の値を 1 つ出力する．以下に，Wave の処理関数のコードを示す．この処理関数は，引数として受け取った amp（振幅）と base（中心点）に基づき，Sin 波に従って，angle（変換元の角度）を数値に変換して返す．

```
function Wave_Wave(node, lc, angle, amp, base)
    return amp * math.sin(angle * math.pi / 180) + base
end
```

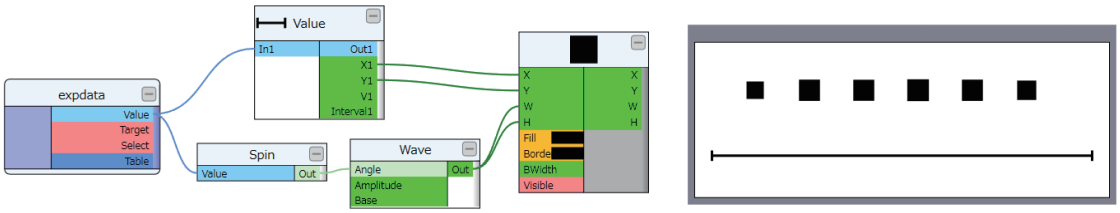
Wave は，Angle，Amplitude，Base の 3 つの入力ピンを持つ．Angle は，変換元の角度を入力するピンのため，接続が必須である．一方，Amplitude と Base は，接続は必須ではなく，未接続の場合には，ノードのパラメータに設定された値が使用される．そのため，開発者は，Iv Studio のノードパネルを用いて，中心点や振幅をインタラクティブに調整できる．

図 6.3 に，Spin と Wave を用いた可視化手法の記述例を示す．まず，データテーブルについて説明すると，データとしては 5 レコードの値を持っており，Value ピンから 0〜3 の数値が出力される．Target ピンは，後に説明する実験タスクにおける，回答の対象を表す真偽値を出力する．図 6.3(a) は，Value の値を Spin アニメーションで表現する可視化手法の例である．Value の値を Spin ノードに入力して角度に変換し，その角度を，楔型（ここでは，楔の角度を鈍角とし，切り込み入りの円を描くように設定されている）を描くノードの角度に入力することで，Spin アニメーションを記述した．図 6.3(b) は，Value の値を Size Vibrate アニメーションで表現する可視化手法の例である．(a) と同様に，Value を角度に変換し，その出力をさらに，Wave を用いて座標空間上の値に変換し，その出力を矩形の幅と高さに入力することで，Size Vibrate アニメーションを記述した．

なお，Spin や Wave のような変換処理は，Iv Studio の演算処理用の雛型を利用することでも記述可能であるが，以降の実験での使用頻度が高いことから，ノードの雛型として実装した．



(a) Spin アニメーション



(b) Size Vibrate アニメーション

図 6.3: Spin アニメーションと Size Vibrate アニメーションの記述例

6.4 実験 1

先に述べたループアニメーションの利点を確認するための評価実験を行った．

6.4.1 実験 1 の概要

ループアニメーションの表現能力を調査する実験を行った．この実験では，以下の 3 点に着目した．

1. ループアニメーションの頻度によって，量的データを表現することができるか？
2. 従来の視覚変数を一つ用いたループアニメーションによって複数の量的データを表現することができるか？
3. ループアニメーションはどのような意味をもったデータに対して適しているか？

量的データの視覚的な表現に期待される性質には様々なものがある．この実験では，量的データを表現できているかどうかについて，量的な値の集合が持つ特徴を表現できるかどうかという側面を調査した．その方法として，ループアニメーションと他の表現とを併用して複数の変量の量的データを同時に提示し，被験者が，その変量間の関係性を読み取ることができるかどうかを調べた．変量間の関係性を読み取るには，例えば，変量 A における集合の特徴と変量 B における集合の特徴の両方の読み取りが必要と考えられる．そのため，この調

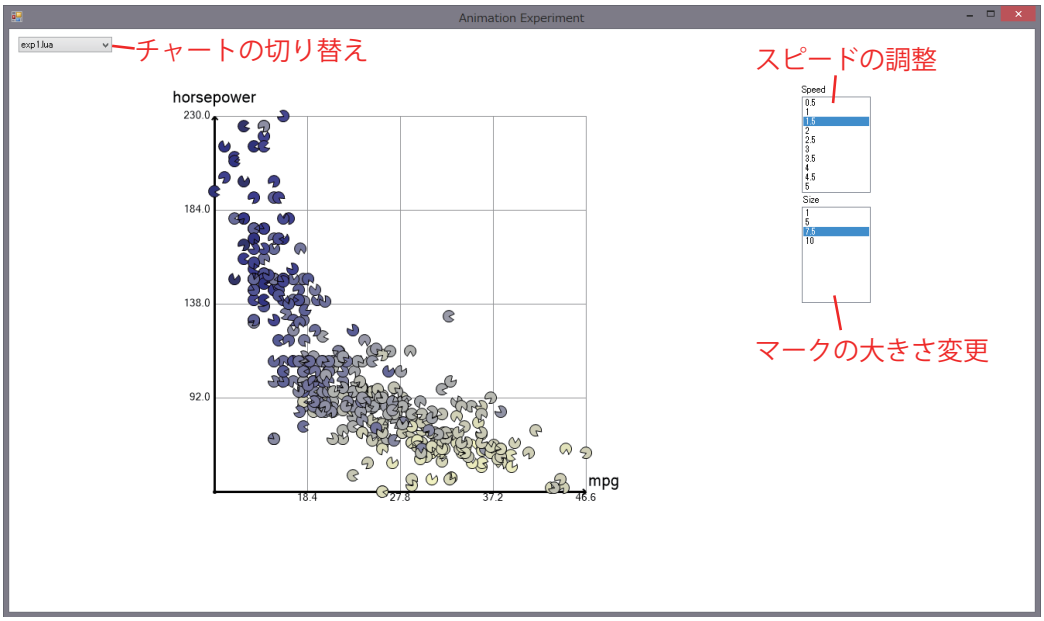


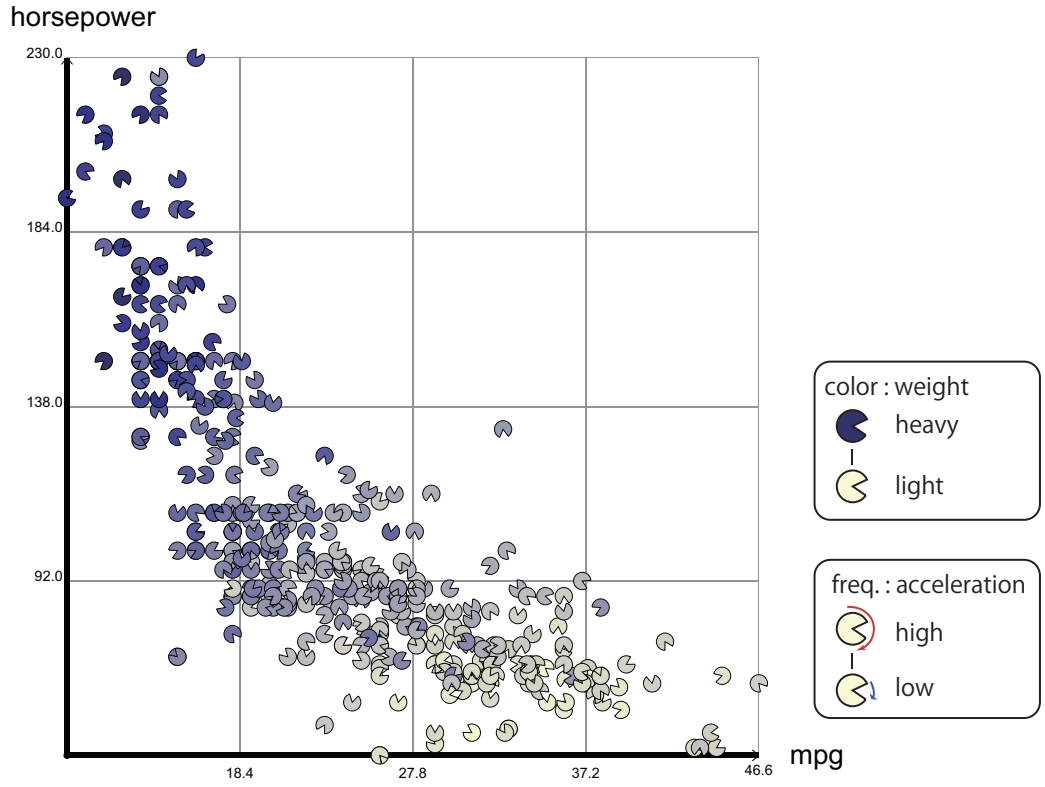
図 6.4: 実験 1 に用いたツール

査によって，ループアニメーションが量的な値の集合が持つ特徴を表現できるかどうかを調査できると考えられる．

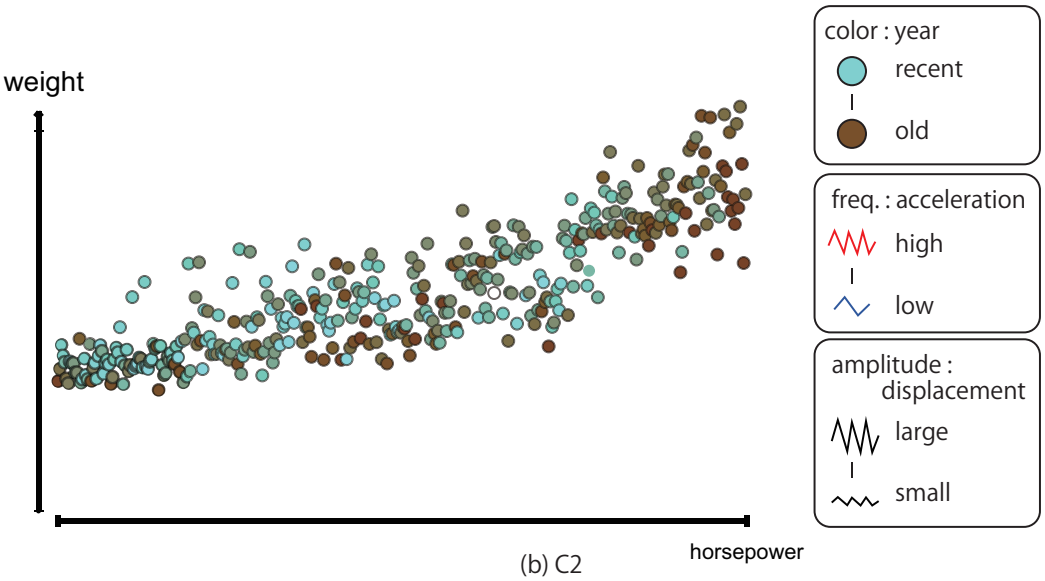
この実験のため，Iv Studio を利用して図 6.4 のような実験ツールを作成した．このツールでは，画面中央にアニメーションするチャートが表示され，ツールの利用者は，画面右側のリストからアニメーションの速度とマークの大きさを調整することができる．また，画面左上のコンボボックスから，表示するチャートを切り替えることができる．

作成したツールでは，3 つのアニメーションするチャートを表示できるようにした．全てのチャートが，自動車のデータ [car] を可視化したものである．図 6.5(a) は，X 軸が燃費（fuel economy），Y 軸が馬力（horsepower），切り込み入りの円の色が車重（weight）を表し，回転の速さが加速性能（acceleration）を表すチャートである．このチャートを C1 と呼ぶ．6.5(b) は，X 軸が馬力（horsepower），Y 軸が車重（weight），円の色が年式（years）を表すチャートである．このチャートでは，各円が垂直方向に振動しており，振幅が排気量（displacement），振動の頻度が加速性能（acceleration）を表している．このチャートを C2 と呼ぶ．もうひとつ，6.5(b) のチャートから加速性能（acceleration）と排気量（displacement）を入れ替えたチャートも作成した．このチャートを C3 と呼ぶ．

6 名の被験者に対して，まず，C1 と C2 を見せ，どのような情報を読み取ることができたかを尋ねた．次に，C2 と C3 を見せ，どちらのチャートが直感的であったかを尋ねた．なお，図 6.5 の右の凡例はツール上では表示されておらず，各チャートの読み方は文章にて説明した．



(a) C1



(b) C2

図 6.5: 評価実験 1 に用いたチャート

6.4.2 実験 1 の結果と考察

C1 から，被験者全員が「馬力の大きい車ほど加速性能が高い」という傾向を読み取ることができた．また，車重よりも馬力のほうが加速性能に与える影響が大きいという傾向も読み取ることができた．C2 からは，被験者全員が「馬力の大きい車ほど加速性能が高く，排気量が大きい」という傾向を読み取ることができた．しかし，C2 から，加速性能と排気量間の関係性に関する回答はなかった．C2 と C3 を比較し，どちらのチャートが直感的であったかを訪ねた質問では，全ての被験者が C2 の方が直感的であったと回答した．

被験者は，C1 から，ループアニメーションが提示する加速性能と他の表現の提示する馬力や車重との間の関係性を読み取ることができた．このことから，ループアニメーションと他の静的な表現を併用することで，2 変量間の関係性を提示できたと考えられる．つまり，ループアニメーションは，量的データの集合が持つ特徴を表現できていたと言える．

C2 では，ループアニメーションは加速性能と排気量の両方を提示していた．被験者が加速性能と馬力間の関係性と，排気量と馬力間の関係性の情報を読み取れたことから，ループアニメーションが，位置（Y 座標）という一つの視覚変数を用いて，少なくとも 2 つの情報を提示できていたと考えられる．しかしながら，頻度（加速性能）と Y 座標（車重）の関係性や，頻度（加速性能）と振幅（排気量）の関係性の読み取りが可能かについてはさらなる調査が必要である．また，振動による情報提示は，頻度と振幅の両方が，マークの移動速度に影響を与えるため，閲覧者がこれらの表現から情報を独立に読み取ることができるかどうかについても，今後の調査が必要である．

C2 と C3 を比較し，全ての被験者が C2 の方が直感的であったと回答したことから，頻度による表現は，排気量という体積に関するデータよりも，加速性能という速度に関するデータの表現に適していたと考えられる．可視化手法では，閲覧者がその可視化結果の提示するデータの意味をイメージしやすいことも重要である．この実験結果から，ループアニメーションの頻度表現は，速度に関するデータの表現に適していると言えそうである．

6.5 実験 2

量的データの表現において，個々の値の表現精度も重要な側面である．そのため，既存の視覚的表現とループアニメーションを精度面で比較する実験を行った．

ある値を表現する視覚的表現を複数の人に見せ，読み取ることができた値のばらつきが少ないとき，その視覚的表現は精度が高いと言える．このような考えに基づき，この実験では，視覚的表現から値を読み取ってもらうタスクを設計し，回答のばらつきから表現精度を評価した．

6.5.1 予備調査

視覚的表現には，使用する図形，色，速度に関して様々なバリエーションが考えられる．そのような違いは，表現精度に影響すると考えられる．しかし，多くバリエーション間の違い

表 6.2: Spin のマークの形状に関する回答

マークの形状	票数
切込み入り円（塗）	5
線（片）	3
線（両）	1
星（枠）	1

を実験で網羅的に調べるためには，被験者に大きな負担を強いることとなる．そこで，既存の視覚的表現と比較するループアニメーションの種類を絞り込むために，以下の 2 つに関する予備調査を行った．

1. Spin で回転させるマークの形状
2. Blink で変化させる色の属性

まず，Spin で回転させるマークの形状の検討のため，図 6.6 に示す 12 種類の図形を回転させる表現を作成し，どの表現が多くの人に気に入られるかを調査した．どの表現が気に入られるかを調査することで，実験時に，被験者にとって読み取りの負担が少ない形状を選ぶことができると考えられる．その調査のため，図 6.7 のように，画面上に 6 つのマークが表示される Web ページを作成した．ループアニメーションは実験 1 のような多数のマークが描かれる環境で利用されることを想定しているため，画面に複数のマークを表示するようにした．被験者には，1，2 番目のマークと比べて，4 から 6 番目のマークのうち，下に丸印の付いているマークがどの程度の値を表現しているかを，画面下のスライダーバーを用いて回答してもらった．スライダーバーの値を調整後，OK ボタンをクリックすると，次々に新しい表現が表示されるようにし，12 種類を 5 回ずつ回答してもらった．その後，最も気に入った図形を回答してもらった．作成した Web ページの URL を研究室内で共有し，回答を募った．

Blink で変化させる色の属性については，図 6.8 のように色相，彩度，明るさの 3 種類のループアニメーションを切り替えられ，ループアニメーションに用いられている以外の色の属性を下のスライダーバーで調整することのできる Web ページを作成し，3 名の被験者に見せながら意見を聞いた．

Spin のマークの形状に関する調査では，10 名から回答が得られた．回答の結果を表 6.2 に示す．最も気に入られた形状は，中を塗りつぶした切込み入りの円であった．次に端点を固定して回転させる線（片）が気に入られた．また，マークの形状に関する以外のコメントとして,「回答するマーク（下に丸印の付いているマーク）が移動すると，誤ったマークについて回答してしまう.」という，タスクの設計に関する意見も得られた．

Blink の調査では，色相を用いたループアニメーションは非常に目が疲れ，読み取りが困難という意見が得られた．彩度と明るさを用いるループアニメーションについては，色味の違い（彩度や色相の違い）によって，見やすさが大きく変わるという意見が得られた．

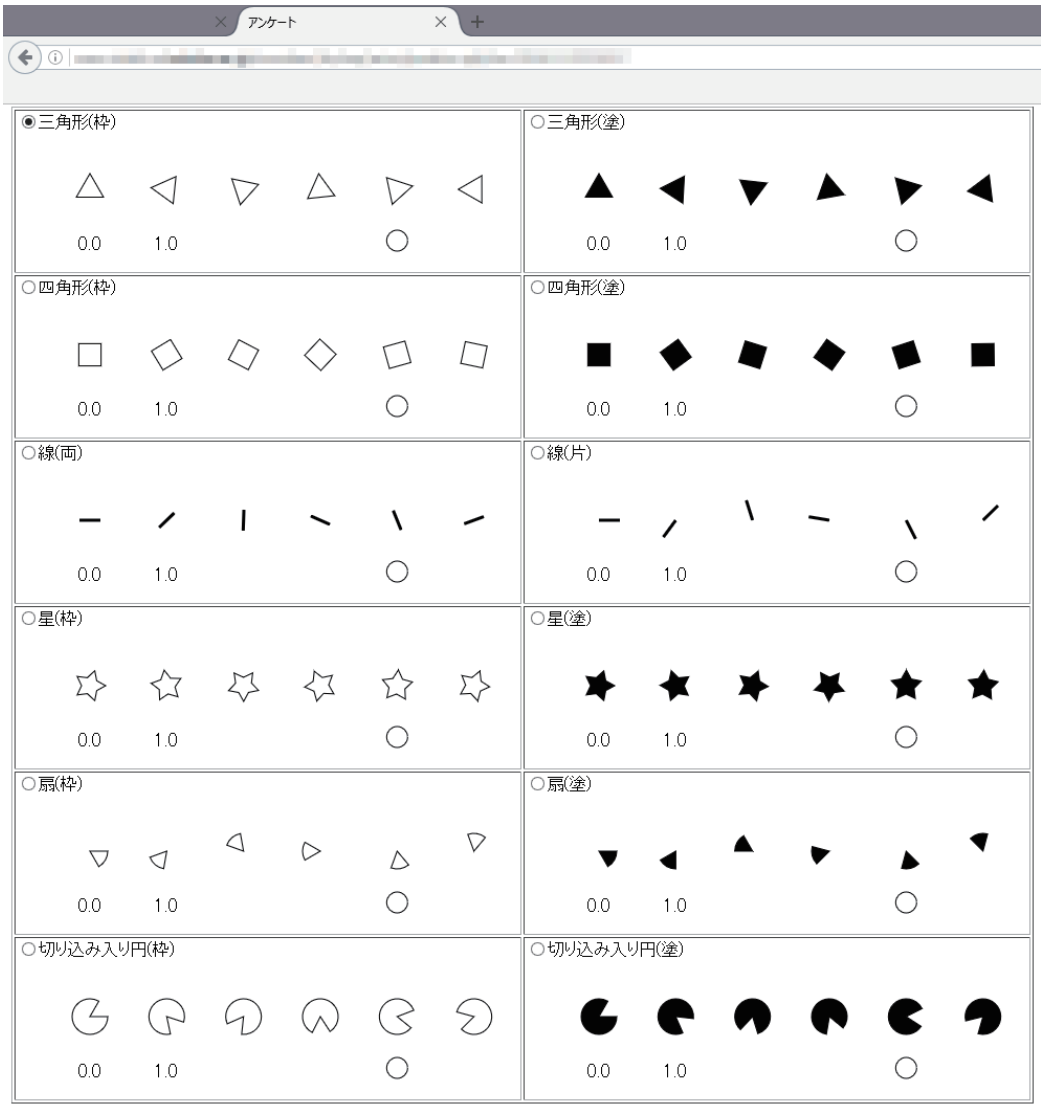


図 6.6: 予備調査で使用した図形（アンケート画面）

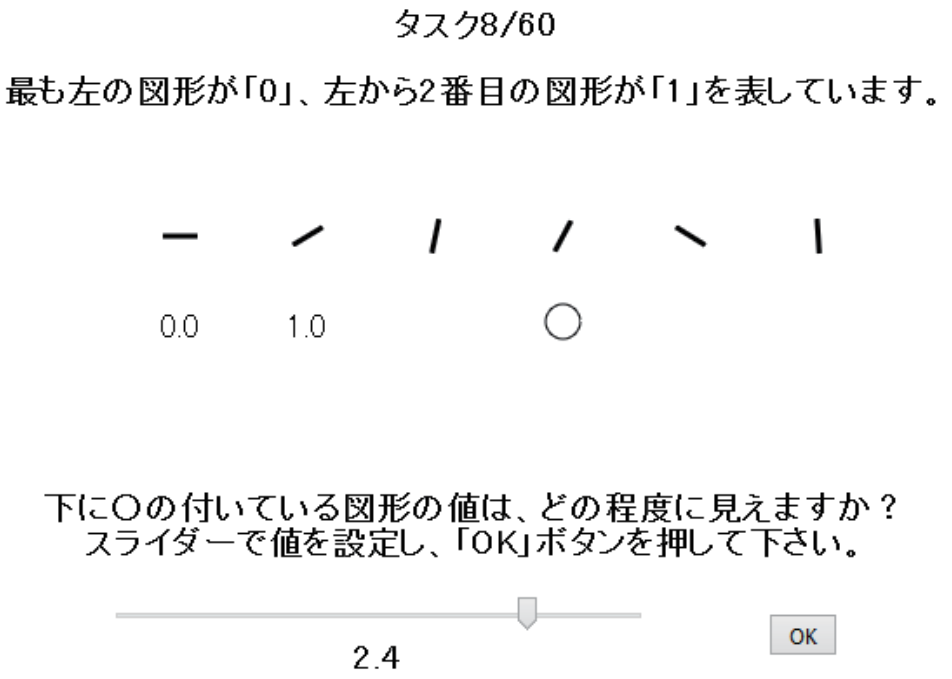


図 6.7: 予備調査のタスク画面

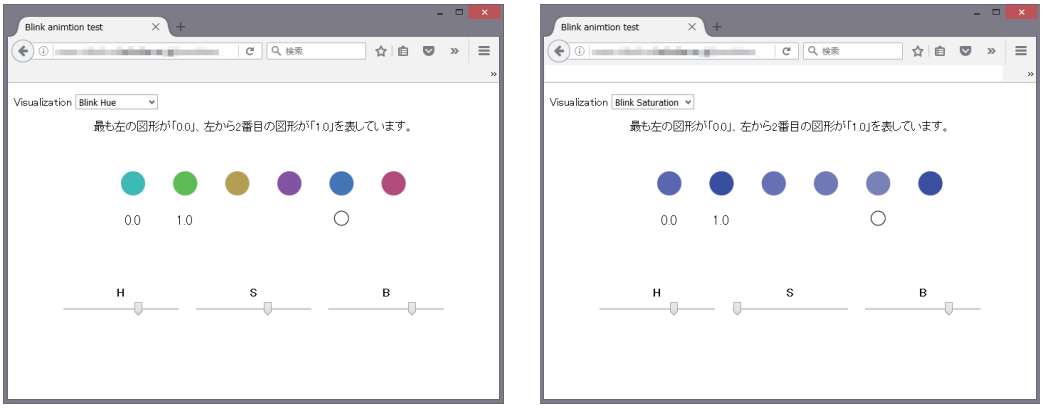
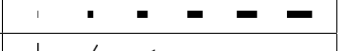
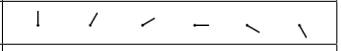


図 6.8: Blink に関する調査用の Web ページ

表 6.3: 実験で用いた静的な視覚的表現

表現	例
Length	
Slope(Tilt)	
Area	
Density(Brightness)	

6.5.2 実験 2 の概要

静的な視覚的表現として，Length，Slope(Tilt)，Area，Density(Brightness) の 4 つの表現を用いた (表 6.3)．表中の「例」の列の図は，左から順に 0, 0.5, 1, 1.5, 2, 2.5 を表す. 静的な視覚的表現については，Cleveland ら [CM84] や Mackinlay[Mac86] によって，量的データ表現のランキングが提案されている．被験者の負担を軽減するために，そのランキングを参考にして比較対象を 4 つに絞った．

ループアニメーションには，予備実験の結果を考慮し，Spin，Blink of Brightness (Value)，Vertical Vibrate，Horizontal Vibrate，Size Vibrate，Shape Morph の 6 つを用いた．表 6.1 の例の赤い枠で囲まれた表現が，実験で使用されたものである．どのアニメーションでも，値 0 は停止を用いて表現した．Spin の場合には値 1 を 2Hz で表現した．それ以外のアニメーションでは，値 1 を 1Hz で表現した．

予備調査の結果から，Spin には切込み入りの塗りつぶした円を使用し，Blink には色味の影響のない無彩色の明るさを変化させる表現を用いた．

実験では，画面に 5 つのマークを表示し，左から 1 番目のマークが 0，2 番目が 1 を表し，3 から 5 番目のマークが 0 から 3 のランダムな値を表すようにした（図 6.9）．予備調査の結果を受け，画面に表示するマークの数を 5 つに減らし，被験者に回答してもらうマークを 4 番目に固定した．

被験者には，1，2 番目のマークと比べて，4 番目のマークがどの程度の値を表現しているかを回答してもらった．回答には，画面下のスライダーバーを用いてもらった．被験者から読み取られた値を「judged value」と呼ぶ．スライダーバーの範囲は 0 から 3 で，0.1 刻みとした．スライダーバーの値を調整後，OK ボタンをクリックして回答を完了してもらった．回答に時間制限は設けず，被験者にできるだけ速く，かつ正確に回答するように求めた．

各視覚的表現を用いたタスクを 1 回ずつ行うものを 1 セットと呼ぶ．1 セット中の視覚的表現の順序は，順序効果を打ち消すために毎回無作為に選んだ．3 番目，4 番目，5 番目のマークが表現しようとする値も 0.2 から 2.8 の間で無作為に選んだ．この値を「given value」と呼ぶ．

タスクの回答方法を被験者に説明した後，被験者に 1 セットの練習を行ってもらった．練習中，視覚的表現の読み取り方について補足説明を行った．例えば，Area については面積で値を表現していることを口頭で補足説明した．練習後，3 セットのタスクを行ってもらい，3 分間の休憩を挟んで，さらに 3 セットのタスクを行ってもらった．つまり，合計 6 セット，合計 60 回のタスクを行ってもらった．実験は，練習を含め 30 分程度で完了することを想定した．

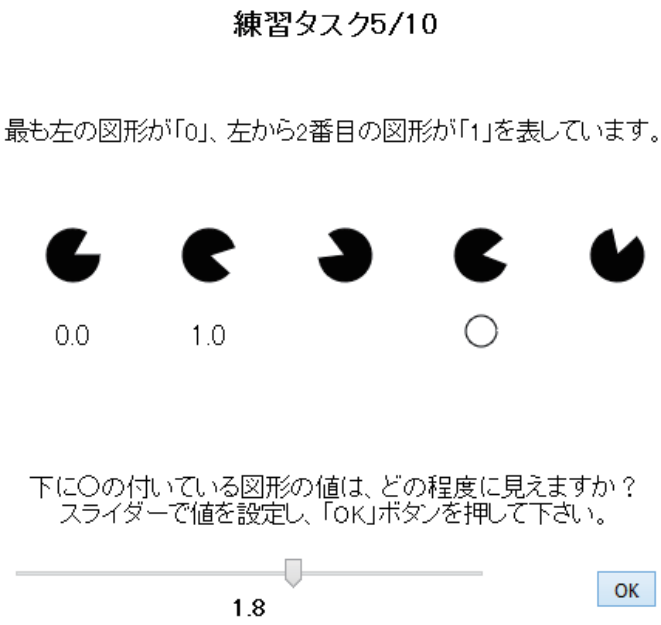


図 6.9: 評価実験 2 のタスク画面

実験のプログラムは，予備実験同様に Web ページ上で実行した．ただし，実験の環境や機材は統一した．全被験者とも同一の PC を使用し，モニタにはカラーマネジメントシステムを搭載した Eizo ColorEdge CG277 を用いた．Web ブラウザには，Windows 版の Mozilla Firefox 42.0 を用い，ブラウザの画面は最大化し，実験画面以外は表示されないようにした．

被験者は，情報科学を専攻する大学生及び大学院生 14 名で，このうち，7 名が情報可視化を専門とする研究室に在籍していた．

6.5.3 実験 2 の結果と考察

図 6.10 に，視覚的表現ごとの実験結果の散布図を示す．実験の回答のうち，1.0 よりも低い値を表現しているマークを 1.0 より大きく答えた回答と，1.0 よりも大きい値を表現しているマークを 1.0 より小さく答えた回答は，比較対象の視覚的表現（左から 2 番目の表現）との大小関係を勘違いしてしまったものと考えられる．そのため，judged value の逆数を，以降の分析に用いる値とした．

ばらつき度合いを算出する方法として，2 つの指標を用意した．これらの指標の値の平均値が小さければ，高い精度の表現と判断する．

一つ目の指標は，given value からの平均二乗誤差に基づいたものである．given value を g，judged value を j としたとき，指標 e_1 は式 6.2 のように求められる．

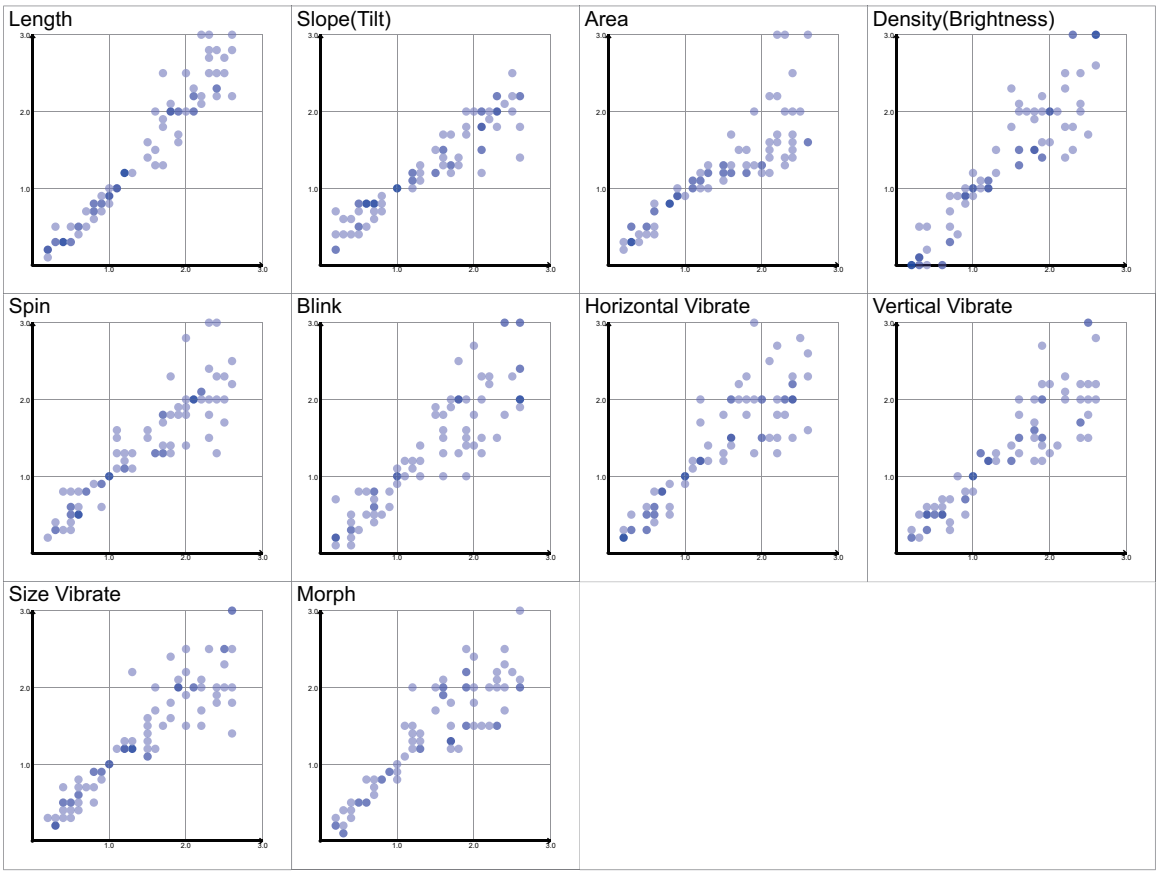


図 6.10: 実験 3 の実験結果

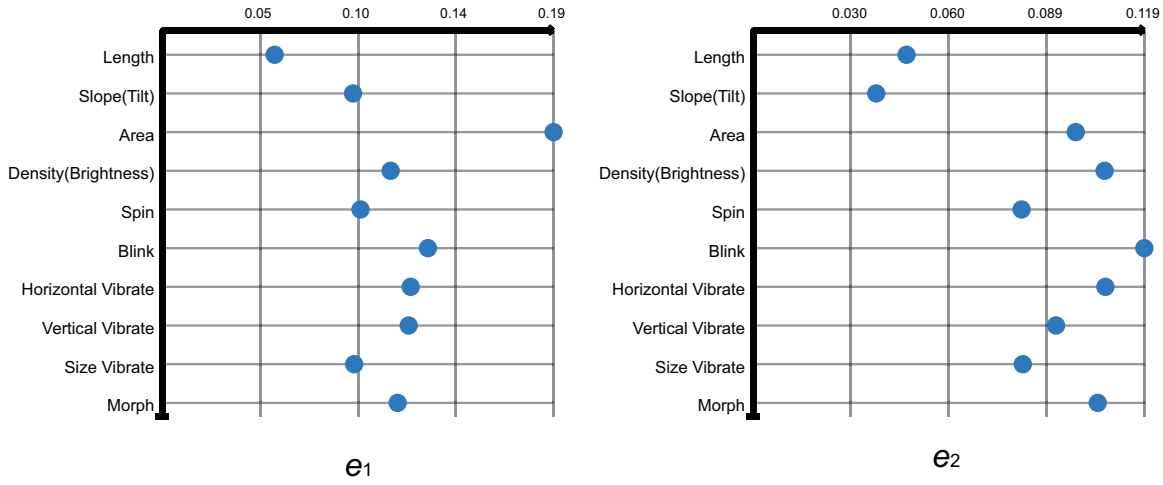


図 6.11: 表現ごとの平均値

$$e_1 = (g - j)^2 \tag{6.2}$$

二つ目の指標は，一つ目の指標に加えて，人の受ける感覚の歪みを考慮した指標である．物理的刺激と人の受ける感覚の強さは，べき乗則になることが知られている（スティーヴンスのべき法則 式 6.3）．

$$E(I) = kI^a \tag{6.3}$$

この法則を考慮し，式 6.4 のように，指標 e_2 を求めた．

$$e_2 = (E(g) - j)^2 \tag{6.4}$$

式 6.3 の係数 k, a には，累乗近似曲線の係数を用いた．

指標 e_1，e_2 の表現ごとの平均値を図 6.11 に示す．平均値が小さいほど，精度が高いことを示している．この結果について，分散分析を行った．e_1 については，有意水準 5%で有意差が認められた（Pr(>F) = 0.00736）．一方，e_2 については有意差が認められなかった（Pr(>F) = 0.0566）．

e_1 について，テューキー・クレーマーの方法を用いた多重比較を行った．有意水準 5%で，以下の表現間に有意差が認められた．

- Length - Area [p = 0.0005080]

given value が 1 以下のタスクと，1 以上のタスクに分けて分析を行った．1 以下のタスクは，基準となるマークの間を分割するタスクであり，1 以上のタスクは，基準からの倍率を読み取るタスクであることから，種類の異なるものと考え，分割して分析した．

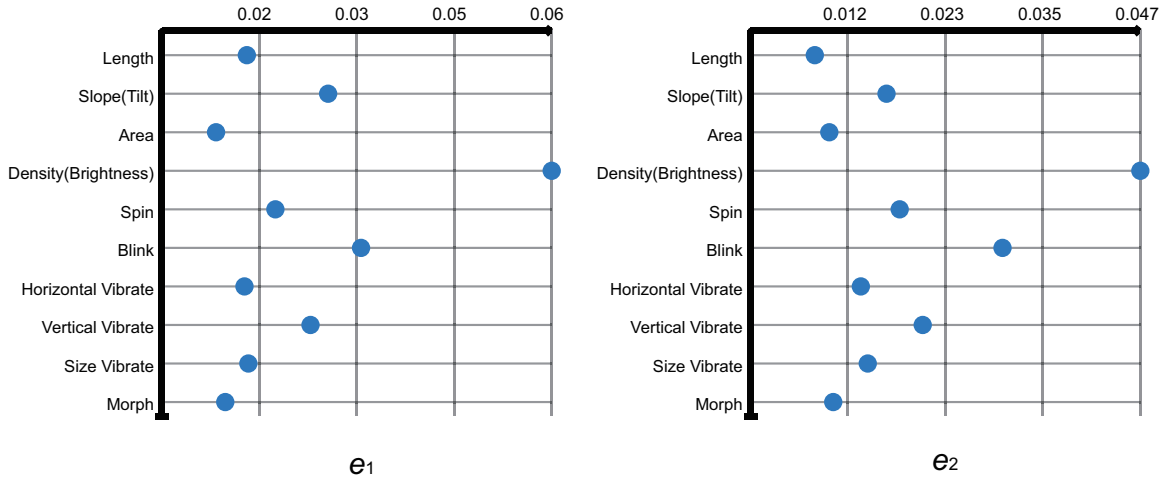


図 6.12: 1 以下のタスクにおける表現ごとの平均値

まず，given value が 1 以下のタスクについて，指標 e_1，e_2 の表現ごとの平均値を図 6.12 に示す．この結果について，分散分析を行ったところ，e_1，e_2 ともに，有意水準 5%で有意差が認められた（e_1 : Pr(>F) = 0.00001 e_2 : Pr(>F) = 0.00002）．

e_1 について，多重比較を行ったところ，有意水準 5%で，以下の表現間に有意差が認められた．

- Length - Density(Brightness) [p = 0.0001785]
- Slope(Tilt) - Density(Brightness) [p = 0.0230185]
- Area - Density(Brightness) [p = 0.0000689]
- Spin - Density(Brightness) [p = 0.0020463]
- Horizontal Vibrate - Density(Brightness) [p = 0.0002112]
- Vertical Vibrate - Density(Brightness) [p = 0.0101406]
- Size Vibrate - Density(Brightness) [p = 0.0002748]
- Morph - Density(Brightness) [p = 0.0002015]

e_2 について，多重比較を行ったところ，有意水準 5%で，以下の表現間に有意差が認められた．

- Length - Density(Brightness) [p = 0.0000396]
- Slope(Tilt) - Density(Brightness) [p = 0.0044267]

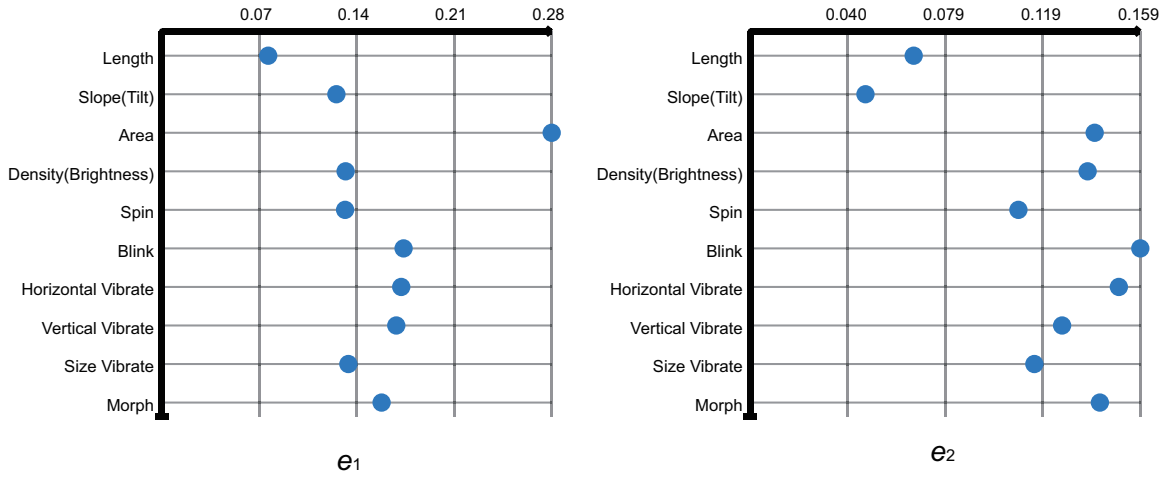


図 6.13: 1 より大きいタスクにおける表現ごとの平均値

- Area - Density(Brightness) [p = 0.0003346]
- Spin - Density(Brightness) [p = 0.0140946]
- Horizontal Vibrate - Density(Brightness) [p = 0.0012259]
- Vertical Vibrate - Density(Brightness) [p = 0.0327212]
- Size Vibrate - Density(Brightness) [p = 0.0018702]
- Morph - Density(Brightness) [p = 0.0006455]

次に，given value が 1 以上のタスクについて，指標 e_1，e_2 の表現ごとの平均値を図 6.13 に示す．この結果について，分散分析を行ったところ，e_1 にのみ，有意水準 5%で有意差が認められた（e_1 : Pr(>F) = 0.00203 e_2 : Pr(>F) = 0.0867）．

e_1 について，多重比較を行ったところ，有意水準 5%で，以下の表現間に有意差が認められた．

- Length - Area [p = 0.0001547]
- Slope(Tilt) - Area [p = 0.0148296]
- Density(Brightness) - Area [p = 0.0247006]
- Spin - Area [p = 0.0228511]
- Size Vibrate - Area [p = 0.0272625]

以上の結果をもとに，静的な視覚変数とループアニメーションの表現精度の差に関して考察した．

まず，全区間の平均値については，静的な視覚変数とループアニメーションを比較すると，Area や Density(Brightness) よりも Spin と Size Vibrate の方が指標の平均値が小さかったものの，その間に有意差は認められなかった．

given value が 1 以下のタスクにおいては，e_1 と e_2 の両指標について，Blink 以外のループアニメーションが，Density(Brightness) よりも精度が高かった．そのため，2 つの表現の間を分割するような利用方法において，ループアニメーションは Density(Brightness) よりも高い精度を持っていたと言える．ただし，この実験では，1 以下の Density(Brightness) のタスクの難易度が高かったことを述べておく．タスクでは，0 から 3 の値を表現するために，比較対象となる 1.0 を表す Density(Brightness) の表現に暗いグレーが割り当てられた．そのため，被験者にとって，黒と暗いグレーの間の判断が難しかったと考えられる．タスクが 0〜2 の値を表現すためのタスクを行った場合，結果が異なる可能性もあるため，今後より詳細な調査が必要である．

given value が 1 以上のタスクにおいては，e_1 に関して，Spin と Size Vibrate が Area よりも精度が高かった．Area は，given value が大きくなるにつれ，人の感じ取る歪みの影響が大きくなるため，この差が現れたものと考えられる．歪みを考慮した e_2 では，平均値は Spin と Size Vibrate の方が Area よりも小さいものの，有意差は認められなかった．今後は，歪みを考慮した際の精度についてより詳細な調査が必要と考えられるものの，この実験により，値をそのまま視覚的属性に線形に割り当てるような素朴な利用方法をする場合においては，Spin と Size Vibrate が Area よりも高い精度で量的データを表現できることが分かった．1 以上のタスクでは，Density(Brightness) と Area の間にも有意差が認められ，Density(Brightness) の方が高精度という結果となった．これは，Mackinlay[Mac86] のランキングと矛盾する．その原因としては，タスクで用いた Density(Brightness) 表現の表現可能な範囲が 0 から 3 であり，スライダーバーで回答可能な範囲と一致していたことから，回答の上限に対してある程度の予測がついてしまった可能性が考えられる．今後は，Density(Brightness) との比較方法を再検討した調査が必要である．

この実験から得られた知見をもとに，Mackinlay のランキングに対してループアニメーションを加えた量的データの表現精度のランキングを作成した（図 6.14）．given value が 1 以上のタスクの結果から，Spin と Size Vibrate を Area の上位に配置した．ただし，このランキングは，少量のデータ提示において，値を視覚的属性に線形にエンコードするような素朴な利用方法を想定したものであり，実験 1 で調べたような集団の持つ特徴の提示や，読み取りにかかる時間などが考慮されたものではない．今後は，多量のデータを提示した際の集団的特徴の提示能力や，値や特徴の読み取りにかかる時間について，さらなる調査が必要である．

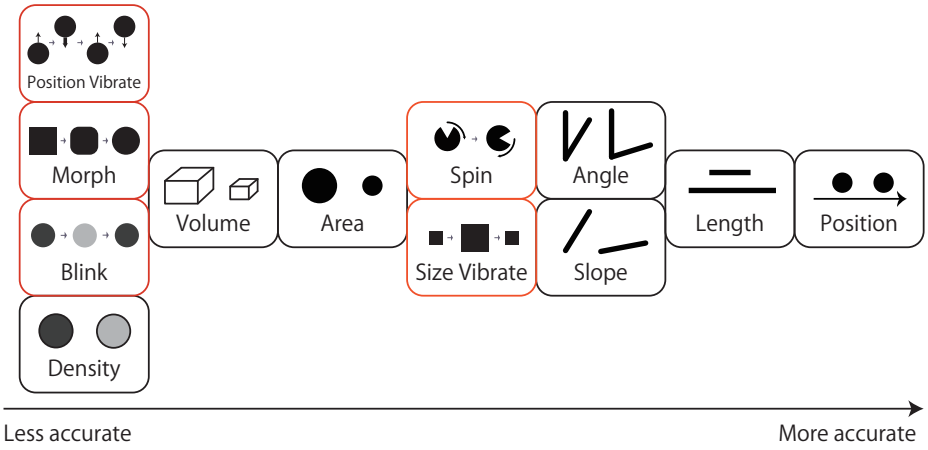


図 6.14: 量的データの表現精度のランキング

6.6 Iv Studio の利用

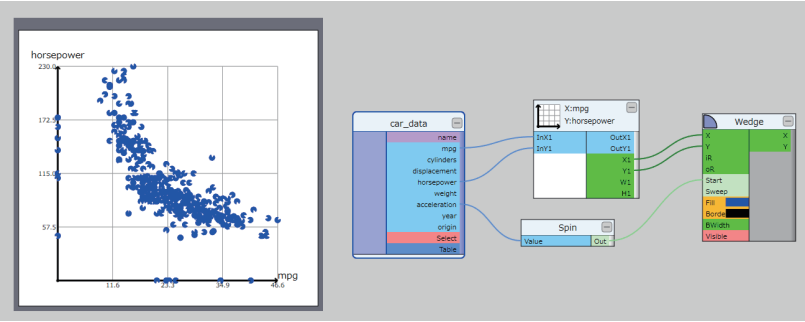
本節では，実験のために作成したプログラムについて，Iv Studio の利用という観点を中心に説明する．そして，Iv Studio の利用に関する考察を述べる．

6.6.1 実験 1 で用いたツールの作成

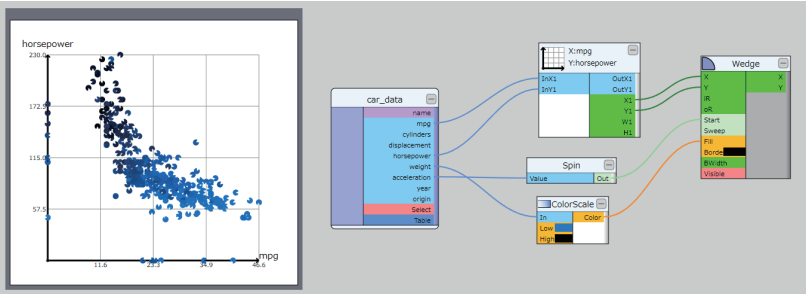
はじめに，実験 1 に至った経緯について説明する．筆者は，6.3 で述べたループアニメーション用のノードの雛型を作成したのち，ループアニメーションの特性を調べるために，Iv Studio 上で図 6.15 のような，Spin アニメーションを用いた複数の可視化手法を記述し，筆者自身で検証を行った．図 6.15(a) が，散布図に Spin アニメーションを加えたもの，(b) が (a) に加えて，色を用いて量的データを表したもの，(c) が色を用いてカテゴリデータを表し，プロットの大きさで別の量的データを表したもの，6.15(d) が，(b) の配色を変え，プロットに枠を付けたものである．図 6.15(d) は，図 6.5(a) と同一のものである．これらの図から，ループアニメーションによって情報を提示することができる可能性を感じ，実験 1 を計画した．

実験 1 で用いたツールの開発は，前段落で述べた作業の延長として，まずは Iv Studio 上で可視化手法の実装を行うところから始めた．6.4.1 で述べた 3 点を調査するために，Iv Studio 上で実験に用いる可視化手法の設計を行った．この設計は，研究室内のメンバーに意見を聞きながら行った．その際に，マークの大きさやアニメーションの速度（Spin ノードの Speed パラメータの値）の好みが人によって異なることが分かった．そのため，アニメーションの速度とマークの大きさについては，Iv Studio 上で微調整を行うのを止め，実験ツール上で調整できるようにすることを決めた．

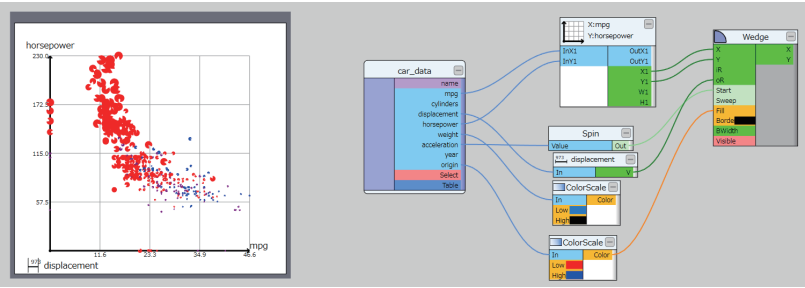
次に，実験ツールの開発を行った．開発言語には，筆者が使い慣れているという理由から，C#を用いた．実験ツールの画面や入力部分を作成し，NLua[NLu] を用いて，Iv Studio で実装した可視化処理を実行できるようにした．一点，開発に困ったことは，Iv Studio の構築時に



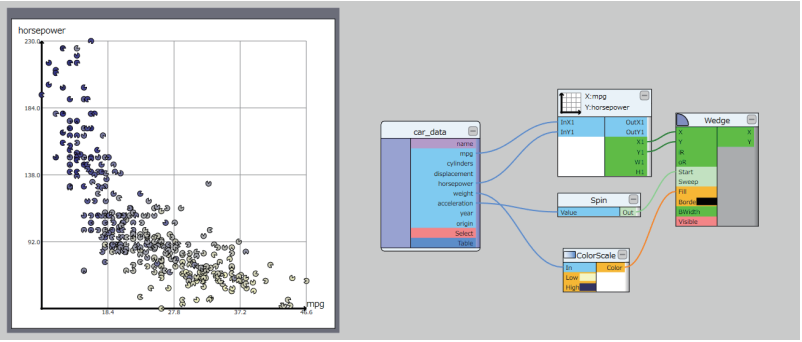
(a)



(b)



(c)



(d)

図 6.15: 実験 1 の前に作成した可視化手法とデータフロー図

は，埋め込み先のプログラムからノードのパラメータを制御することを想定していなかったことである．しかし，Iv Studio からの出力形式が，Lua のスクリプトになっていたため，Lua のインタフェースを介して値を設定することで，Iv Studio の改修を必要とせずに実験ツールを開発することができた．

6.6.2 実験 2 で用いたツールの作成

実験 2 で用いたツールは，6.5.1 で述べた予備実験の，Web を通して回答を集めるという実験方法から，Web ベースで動作することが必須の要件となっていた．そのため，可視化処理の実装言語は JavaScript に決まった．また，使用する可視化手法よりも先に，実験のタスクが決まったため，まず，タスクのページの表示や，データ収集のための，サーバサイドの PHP のプログラムを開発した．そして，Iv Studio を利用して，可視化手法の実装や選定を行い，タスクのページに埋め込むための可視化処理を開発した．予備実験に用いた可視化手法の一部のデータフロー図を図 6.16 に示す．図 6.16(a) が，図 6.6 における線（方），(b) が三角形（塗），(c) が扇（塗），(d) が切り込み入り円（塗）を記述したものである．Iv Studio には，三角形の描画ノードがなかったことから，(b) のようにベクトル演算のノードを用いて三角形の描画を作成した．(c) と (d) は，どちらもデータフロー図に違いはないが，楔型を描く Wedge ノードのパラメータの設定を変えることで，扇と切り込み入り円の両方を記述した．これらのデータフロー図は一見複雑に見えるが，(a) の赤枠内のデータフロー図は，表現の下の 0.0 や 1.0，丸印を表示するための処理であり，可視化手法自体の記述はそれ以外である．赤枠内では，演算処理のノードを利用して，Value が 0.0 と 1.0 のときのみ文字を表示し，Target ピンが True の表現の下にのみ丸印を表示する処理が記述されている．

実験 2 の 6.5.2 で述べた本実験のツールに関しては，Web ベースである必要はなかったが，予備実験のものを再利用した方が効率的であったため，引き続き Web ベースのツールとした．可視化手法については，研究室内のメンバーに意見を聞きながら大きさの調整などを行った．Morph のアニメーションだけは，Iv Studio に用意された雛形だけではデータフロー図の記述が複雑になることから，値によって角の丸みを制御できる丸角矩形のノードの雛型を作成して対応した．

実験 2 では，実験結果の分析を支援するために，実験用ツールの他に，実験結果を表示するページも作成した．サーバサイドのプログラムとして，実験結果を CSV 形式で出力するものを実装し，その出力結果を入力として，視覚的表現ごとの散布図を表示する Web ページを作成した（図 6.17）．ここでの可視化処理も，Iv Studio を用いて作成した．この研究のような実験では，実験途中の経過を観察したいことがよくある．作成した実験結果の可視化ページは，途中経過を手軽に確認することができたため，実験を進めていく中で有用であった．

6.6.3 Iv Studio の利用に関する考察

情報可視化の研究への Iv Studio の利用を通して，Iv Studio が，プロトタイプ制作と実システムへの実装を，実際に支援することができることを確認した．以降，Iv Studio の利点と，利

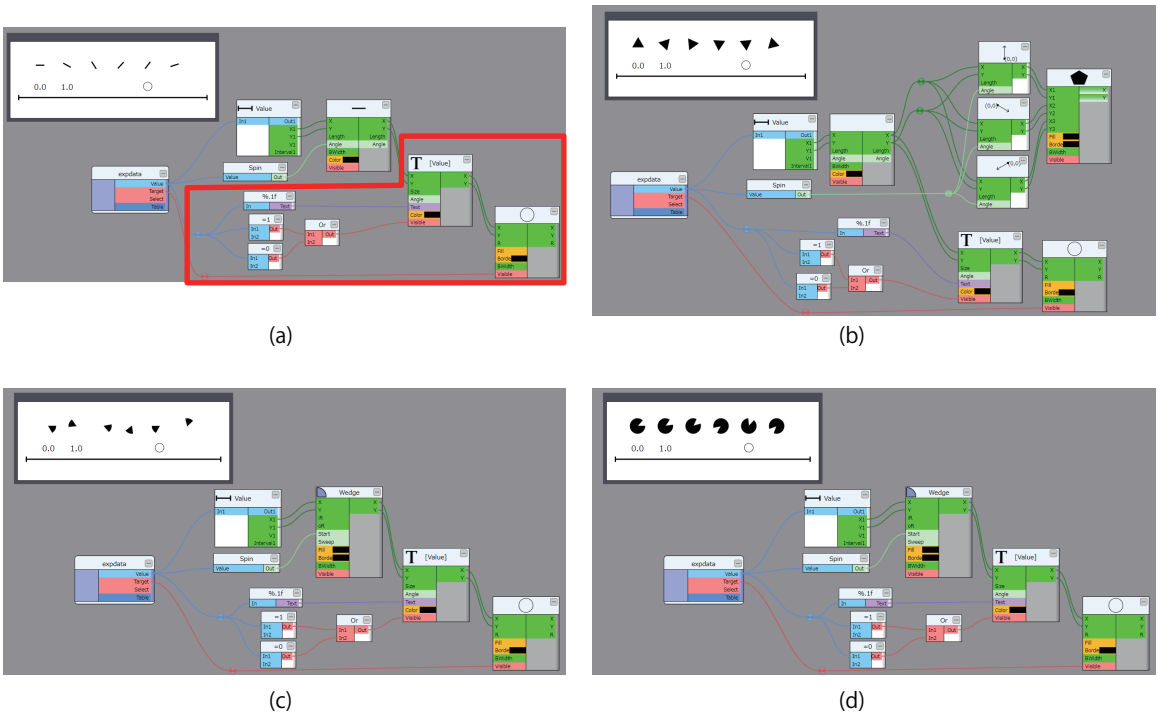


図 6.16: 実験 2 の予備実験で用いた可視化手法とデータフロー図

用を通して見つかった課題について述べていく．

実験に用いたツールの開発のうち，可視化処理部分については Iv Studio のみで担うことができた．6.3 で述べたループアニメーション用のノードの雛型の作成など，一部プログラミング言語を用いた拡張が必要ではあったが，大半の記述は，データフロービジュアル言語のみで完了した．このことは，Iv Stuido が，多様な可視化手法の記述や，拡張性を有していたために可能であったと考えられる．ノードの雛形の拡張は，数行のコードを記述すればよく，その作業自体は数十分で完了した．それよりも，このような実験ツールの開発では，そのノードを使って，どのような可視化手法を作成するのかといった，手法の設計の試行錯誤に多くの時間を要するものであり，Iv Studio は，その作業に役立った．図 6.15 で示した可視化手法の作成は，まさにこの事例であり，様々な応用方法を，実際に作りながらすぐに検証していけることが，Iv Studio の大きな利点である．ただし，Iv Studio を利用した場合と，利用しなかった場合とで，どの程度効率性に差があるのかといった定量的な結果を示すものではないため，今後調査が必要である．

Iv Studio 上で実装した処理が様々なプログラムに埋め込み可能という特徴によって，実験ツールに用いる開発言語に関して大きな制約を課す必要がなかった．例えば，実験 1 では，筆者の慣れた言語を用いることができ，実験 2 では，Web ページ上で動作させるという要件に対応することができた．開発に Iv Studio を導入しても，プログラミングに使用する環境に大きな制約を課さないということも，Iv Stuido の利点である．

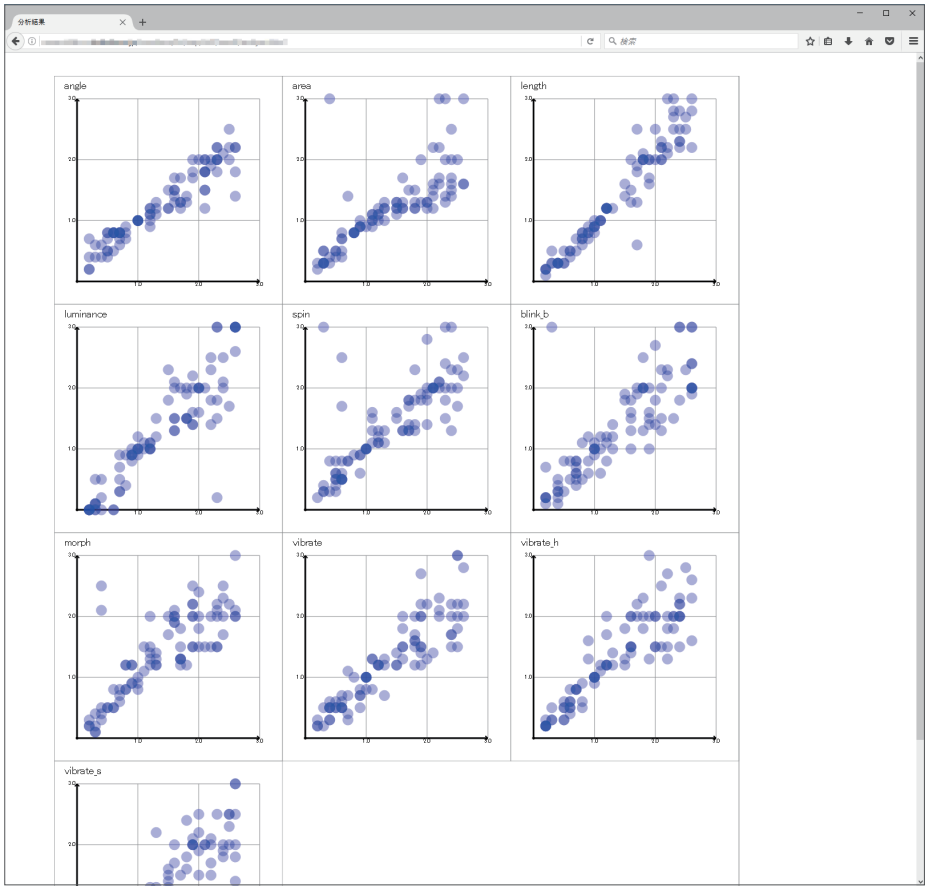


図 6.17: 実験結果可視化用の Web ページ

開発のプロセスに着目すると，実験 1 のように，ひとまず Iv Studio で可視化手法の設計の試行錯誤を行い，その後，実験ツールを開発するというプロセスは，ビジュアルツールを使った有効な場合の開発プロセスの一つと考えられる．特に，Iv Studio を使った場合には，後の実装の心配をせずに，可視化手法の設計の試行錯誤に集中できる．可視化手法の設計が完了したのちに，実験に必要なツールを開発すれば良く，そのツール開発に用いる環境も，実験方法や実験者の好みで自由に選択することができる．従来の，例えば，プログラミング言語のツールキットを使った開発では，後の実験ツールのことを考えて言語を選択するか，設計の試行錯誤に合わせて，後の実験ツールの開発言語を決定する必要がある．一方，Iv Studio を用いた開発では，設計の試行錯誤と後の実験ツールの実装を分けて考えることができる．この開発プロセスは，この研究のような実験ツールに限らず，実システムにおいても同じことが言えよう．このことも，Iv Studio の大きな利点である．

Iv Studio を利用する中で課題も見つかった．実験 1 用のツールのように，可視化処理を利用する場合には，ノードのパラメータをツールのプログラムから制御できる必要があることが分かった．このことに関しては，Lua を出力形式としていたことで対応はしていたが，パ

ラメータ制御のためには，少々煩雑なコードの記述が必要であったため，この点は改良の必要がある．

図 6.16(a) の赤枠内の記述のような，条件演算などは，データフロー図が煩雑になるため，Iv Studio 上での記述には向かない処理である．このような処理は，ノードの雛形の拡張機能を利用してプログラミング言語を使った記述も可能ではあるが，雛形の定義などの記述が必要である．実験ツールの開発では，雛形の記述に比べれば，データフロー図を描いたほうが速いと判断したため，データフロー図で記述したが，処理部分だけを Lua を使って記述できるような機能があれば，そちらを選択していたと考えられる．この研究への利用を通して，必要な処理が実装できないことから実験ツールの作成に利用できないといったことを避けるためにも，演算処理のノードを備えることは重要であることが分かったと同時に，このような処理の記述をより簡単にする機能も重要なことが分かった．そのため，一時的な処理をプログラミング言語で手軽に書くことのできるような機能拡張が必要と考えられる．

マークの記述についても，データフロー図が複雑になることは課題である．図 6.16(b) の三角形の記述は，容易に記述できたとは言い難い．また，必ずしもプログラミング言語であれば容易に実装できるものとも言えない．このような，マークの形状に関しては，例えば，石川ら [石川 14] のような，別のデザインインタフェースが必要になると考えられる．この点については，単純な機能拡張だけではなく，今後のユーザインタフェース研究が求められる．

第7章　結論

本研究では，情報可視化の開発における，プロトタイプ制作と実システムへの実装の両方を支援する開発環境を構築し，その有用性を確認した．

構築した開発環境 Iv Studio のユーザインタフェースには，情報可視化手法の記述のためのデータフロービジュアル言語を採用した．データフロービジュアル言語としては，データの種類とマークの持つ視覚的属性の間を網羅するように変換方法を用意することで，多様な情報可視化手法を簡潔に記述できるようにした．

実システムへの埋め込みを容易にするために，組み込み用スクリプト言語による実行環境を設計し，その実行環境を埋め込む際に必要な実装の手間を少なくするようなインタフェース関数を設計した．そして，構築した開発環境に，その組み込み用スクリプト言語のソースコードを出力する機能を加えた．

筆者は，ビジュアルインタフェースを用いた情報可視化手法の記述が今後主流になると考えている．これらの成果は，実システム開発への利用を見据えたビジュアルインタフェースを備えた情報可視化の開発環境を構築する際の，ユーザインタフェースの設計や，様々な利用先への対応に関する知見となると考えられる．

Iv Studio を，ループアニメーションによる量的データの表現に関する研究に利用した．その研究への利用は，プロトタイプ制作と実システムへの実装の両方を支援することが，情報可視化研究の促進につながることを示した実例である．

7.1　本研究の貢献

情報可視化分野における貢献とデータフロービジュアル言語分野における貢献について，筆者の主張を述べる．

7.1.1　情報可視化分野における貢献

本研究の情報可視化分野における貢献は以下の 4 点である．

情報可視化の活躍の場の拡大

構築した開発環境は，情報可視化のプロトタイプ制作と実システムへの実装の両方を支援することで，より多くの実システムで情報可視化を利用しやすくした．つまり，情報可視化の活躍の場を広げた．本研究では，システムに可視化機能を追加する開発や，情報可視化研

究に用いる実験ツールの開発において，構築した開発環境を実際に用い，それらの開発を支援できることを確かめた．

情報可視化手法の記述のためのユーザインタフェースの改良

本研究では，可視化手法の記述方法として，今後，ビジュアルインタフェースが主流になると考え，既存のビジュアルインタフェースを改良するために，従来形式のユーザインタフェースがデータの流れを開発者に提示していないという点に着目し，情報可視化手法の記述のためのデータフロービジュアル言語と，その言語を用いたユーザインタフェースを開発した．被験者実験により，開発したユーザインタフェースが，データの流れを開発者に提示しない従来形式のユーザインタフェースと比べ，理解のしやすさと操作効率を向上させたことを示した．

埋め込みが容易な可視化の実行環境の設計

プログラミング言語，描画 API，データ形式といった，情報可視化のプログラムが依存する環境を考慮し，様々なプログラムに対して，ビジュアルツール上で実装した可視化処理を埋め込むことができる，可視化の実行環境を設計し，その具体的な実装を示した．埋め込みの際にかかる開発者の負担を軽減するような，可視化処理の実行環境の選定と，インタフェース関数の設計を行った．このような実行環境の設計により，ビジュアルツール上で実装した可視化処理を様々なプログラムから利用できるようにした．

量的データ提示のための時間軸の利用方法の提案

量的データを提示することのできる視覚的属性の種類を増やすため，ループアニメーションを提案した．そして，ループアニメーションのバリエーションを提案し，頻度による量的データの表現精度の調査を行った．その結果から，図形を回転させるアニメーションや図形の大きさを変化させるアニメーションが，比較的高精度に量的データを表現できることを示した．

7.1.2 データフロービジュアル言語分野における貢献

本研究では，情報可視化手法の記述のためのデータフロービジュアル言語を開発し，データフロービジュアル言語の応用先を広げた．

多様な情報可視化手法の記述を可能とするため，手法の記述のためのノードの雛形のライブラリを整備した．従来の可視化のためのデータフロービジュアル言語としては，可視化ビューをノードとして提供するものや，科学的可視化のための構築部品を提供するものがあったが，多様な情報可視化手法を記述することはできなかった．そこで，本研究では，現在，情報可視化では主流の，データ，視覚値への変換方法，マークという 3 要素に着目したライブラリを整備することで，従来よりも多様な可視化手法を記述できるようにした．そして，被験者実験により，データのフローを利用者に視覚的に提示するという，データフロービジュアル言語が本来持つ特徴が，情報可視化手法の記述において有用であることを示した．

7.2 今後の課題と展望

最後に，本研究を進めた中で見つかった課題や，今後の展望を述べる．

7.2.1 情報可視化手法記述のためのビジュアルインタフェースに対する課題と展望

本研究では，従来形式のユーザインタフェースがデータの流れを開発者に提示していないという点に着目し，データフロービジュアル言語をユーザインタフェースに採用した．開発したデータフロービジュアル言語は，多様性と拡張性を考慮し，ビジュアル言語のスタイルとしては，汎用システム向けのものを採用した．しかし，ビジュアル言語のスタイルについては，情報可視化手法の記述により適したものに改良する余地があると考えている．例えば，直交座標軸のノードでは，入出力のピンが InX，InY と縦に並んでいるだけであるが，描かれる軸との対応がよりとりやすいデザインに改良することが可能と考えられる．また，本研究では，連結系の表現形式をとったが，例えば，軸とマークの関連性をより強調するために，座標に関する接続関係は領域系の表現を用いるといった表現形式の改良も考えられる．このような，情報可視化手法の記述に適したデータフロービジュアル言語のスタイルについて，今後，更なる研究が望まれる．

本研究では，ノードの雛形のライブラリを整備することで，多様な手法の記述を可能とした．しかし，4.7.4 にて述べた通り，整備したライブラリだけでは記述できない手法が存在する．不足するノードの雛形を補うことも課題であるが，情報可視化手法の記述に関して，どのようなノードの雛形を用意することで，どのような可視化手法が記述可能になるのか，つまり，機能と記述可能な可視化手法の範囲に関するモデルの構築についても，今後の研究が必要な課題である．

構築した開発環境では，データフロー図を編集すると即座に結果が反映される．これは，大きな利点である．その一方で，可視化手法の記述途中では，不完全な可視化結果が画面に表示されることとなる．本研究で行った被験者実験の中で，そのような不完全な可視化結果に戸惑うようなしぐさを見せる被験者もいた．可視化手法の記述にビジュアルインタフェースを用いる際に，記述途中の状態ではどのような可視化結果を見せるべきか，見せないほうがよい状況があるのか，今後の研究が望まれる．また，手法が複雑になった場合，当然ながら記述を間違えることがある．その際の，デバッグ方法や，デバッグに用いることのできるビジュアルな表現方法の研究も，今後望まれる．

4.6 の調査から，同じ可視化手法でも開発者によって変換規則のイメージが異なることや，手法によっても変換規則のイメージが変わるということが分かった．ビジュアルインタフェースの場合，プログラミング言語と比べると，手法の記述の仕方に制約が付きやすいと考えられる．そのため，開発者のイメージの違いに対応したビジュアルインタフェースを構築するために，今後，開発者や手法による変換規則のイメージのバリエーションに関する研究が望まれる．

7.2.2 情報可視化の実行環境に対する展望

本研究では，埋め込みを容易にするという目標のもと，可視化処理の実行環境の選定や，インタフェース関数の設計を行った．今後は，この設計を利用することで，幅広い範囲のプログラムから利用できる情報可視化の開発環境の構築が期待できる．また，提案したインタフェース関数の設計はそのままに，可視化処理の実行環境の実装を変更することで，情報可視化処理のパフォーマンスを向上させることも考えられる．Iv Studio では，ノードの雛形の拡張性や様々な環境での動作を優先して Lua を採用したが，利用可能な言語の代わりに高いパフォーマンスで動作させることを優先して，例えば，OpenCL をベースとした実行環境を開発することで，大規模データを高速に可視化するシステムを構築するといった別の応用方法も考えられる．

謝辞

本論文の執筆において，指導教員である三末和男教授には，多くのご助言を頂きました．学部 4 年生からの 10 年間に渡り，研究の進め方や論文の執筆方法など，丁寧なご指導を頂いたことにより，本論文をまとめるに至りました．筑波大学の名誉教授である早稲田大学大学院情報生産システム研究科の田中二郎教授には，研究内容や研究発表の方法など，多数のご助言をいただきました．インタラクティブプログラミング研究室の高橋伸准教授，志築文太郎准教授，嵯峨智准教授には，ゼミを通じて多くのご助言を頂きました．また，Simona Vasilache 助教には，英文論文の執筆に関して多くのご助言を頂きました．

本論文の審査にあたり，三末和男教授と共に審査委員を務めていただきました，亀山幸義教授，葛岡英明教授，志築文太郎准教授，金尚泰准教授には，本論文をまとめるにあたって有用なご助言，ご指摘を頂きました．

ソフトイーサ株式会社の登大遊氏をはじめとした皆様には，研究に従事することを承諾していただきました．そのおかげで，研究活動を続けることができました．

所属研究室であるビジュアリゼーションとインタラクティブシステム研究室の学生の皆様，インタラクティブプログラミング研究室の学生の皆様，および OB・OG の皆様には，日頃の研究生活のご協力とご支援を頂きました．

家族や，友人には，精神面を支えていただきました．また，未踏事業の OB・OG の皆様には，研究を進めるうえでの様々な刺激を頂きました．

以上の方々に，心より感謝申し上げます．ありがとうございました．

参考文献

[Ang] AngelScript - AngelCode.com. http://www.angelcode.com/angelscript/.

[APP11] Daniel Archambault, Helen C Purchase, and Bruno Pinaud. Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17, No. 4, pp. 539–552, 2011.

[bbp] 1988. Data expo.. ASA Statistics Computing and Graphics. http://stat-computing.org/dataexpo/1988.html.

[Ben98] Ross Bencina. Oasis Rose the composition - real-time DSP with AudioMulch. In *Synaesthetica '98: Proceedings of the Australasian Computer Music Conference*, pp. 85–92, 1998.

[Ber84] Jacques Bertin, editor. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1984.

[BH09] Michael Bostock and Jeffrey Heer. Protovis: A Graphical Toolkit for Visualization. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 15, No. 6, pp. 1121–1128, 2009.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17, No. 12, pp. 2301–2309, 2011.

[car] 1983 ASA Data Exposition dataset. http://stat-computing.org/dataexpo/1983.html.

[cha] キャラミん ストア. https://www.charamin.com/store/.

[Cha06] Wing-Yi Chan. A Survey on Multivariate Data Visualization. *Technical Report*, 2006.

[cli] 共通言語ランタイム (CLR) . https://msdn.microsoft.com/ja-jp/library/cc825639.aspx.

[CM84] William S. Cleveland and Robert McGill. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, Vol. 79, No. 387, pp. 531–554, 1984.

[CMS99] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., 1999.

[CvW11] Jarry H. T. Claessen and Jarke J. van Wijk. Flexible Linked Axes for Multivariate Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17, No. 12, pp. 2310–2316, 2011.

[Ead84] Peter Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, Vol. 42, pp. 149–160, 1984.

[EDF08] Niklas Elmqvist, Pierre Dragicevic, and Jean-Daniel Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 14, No. 6, pp. 1141–1148, 2008.

[ele] Electron - Build cross platform desktop apps with JavaScript, HTML, and CSS. http://electron.atom.io/.

[ems] emscripten. http://emscripten.org.

[Fek04] Jean-Daniel Fekete. The InfoVis Toolkit. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pp. 167–174, 2004.

[fla] Flare — Data Visualization for the Web. http://flare.prefuse.org/.

[Fry08] Ben Fry. ビジュアライジング・データ —Processing による情報視覚化手法. オライリージャパン, 2008.

[GE08] Mark Giereth and Thomas Ertl. Design Patterns for Rapid Visualization Prototyping. In *Proceedings of the 2008 12th International Conference Information Visualisation*, IV '08, pp. 569–574, 2008.

[GGL+14] Samuel Gratzl, Nils Gehlenborg, Alexander Lex, Hanspeter Pfister, and Marc Streit. Domino: Extracting, Comparing, and Manipulating Subsets Across Multiple Tabular Datasets. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, No. 12, pp. 2023–2032, 2014.

[GHL15] Hua Guo, Jeff Huang, and David H Laidlaw. Representing Uncertainty in Graph Edges: An Evaluation of Paired Visual Variables. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 21, No. 10, pp. 1173–1186, 2015.

[gra] GraphEdit の概要. https://msdn.microsoft.com/ja-jp/library/cc370419.aspx.

[HA06] Jeffrey Heer and Maneesh Agrawala. Software Design Patterns for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 5, pp. 853–860, 2006.

[Hae88] Paul E. Haeberli. ConMan: A Visual Programming Language for Interactive Graphics. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pp. 103–111, 1988.

[HCL05] Jeffrey Heer, Stuart K. Card, and James A. Landay. Prefuse: A Toolkit for Interactive Information Visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pp. 421–430, 2005.

[HG05] Daniel E. Huber and Christopher G.Healey. Visualizing data with motion. In *Visualization, 2005.VIS 05.IEEE*, pp. 527–534, 2005.

[Hil91] Daniel D. Hils. Datavis: A Visual Programming Language for Scientific Visualization. In *Proceedings of the 19th Annual Conference on Computer Science*, CSC '91, pp. 439–448, 1991.

[HR07] Jeffrey Heer and George Robertson. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 13, No. 6, pp. 1240–1247, 2007.

[HSMT06] Tomoyuki Hansaki, Buntarou Shizuki, Kazuo Misue, and Jiro Tanaka. FindFlow: Visual Interface for Information Search Based on Intermediate Results. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation*, APVis '06, pp. 147–152, 2006.

[IM16] Takao Ito and Kazuo Misue. A visualization technique using loop animations. In *Human Interface and the Management of Information: Information, Design and Interaction - 18th International Conference, HCI International 2016, Proceedings, Part I, LNCS 9734*, pp. 136–147, 2016.

[Ins85] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, Vol. 1, No. 2, pp. 69–91, 1985.

[JHM04] Wesley M. Johnston, J. R. Paul Hanna, and Richard J. Millar. Advances in Dataflow Programming Languages. *ACM Computing Surveys*, Vol. 36, No. 1, pp. 1–34, 2004.

[JNI] Java SE 7 Java Native Interface 関連の API および開発者ガイド. http://docs.oracle.com/javase/jp/7/technotes/guides/jni/.

[Kei02] Daniel A. Keim. Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 8, No. 1, pp. 1–8, 2002.

[KST09] Atsutomo Kobayashi, Buntarou Shizuki, and Jiro Tanaka. ImproV: A System for Improvisational Construction of Video Processing Flow. In *Human-Computer Interaction. Interacting in Various Application Domains - 13th International Conference, HCI International 2009, Proceedings, Part IV, LNCS 5613*, pp. 534–542, 2009.

[Lab] LabVIEW. http://www.ni.com/labview/.

[luaa] lua.vm.js. https://daurnimator.github.io/lua.vm.js/lua.vm.js.html.

[luab] The LuaJIT Project. http://luajit.org/.

[luac] The Programming Language Lua. https://www.lua.org/.

[Mac86] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, Vol. 5, No. 2, pp. 110–141, 1986.

[med] Olympic medal winners: every one since 1896 as open data — Sport — theguardian.com. https://www.theguardian.com/sport/datablog/2012/jun/25/olympic-medal-winner-list-data.

[Mun14] Tamara Munzner. *Visualization Analysis and Design*. A K Peters/CRC Press, 2014.

[NCS02] Chris North, Nathan Conklin, and Varun Saini. Visualization schemas for flexible information visualization. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, pp. 15–, 2002.

[NLu] GitHub - NLua/NLua: NLua is the bind between Lua world and the .NET world. https://github.com/NLua/NLua.

[NS00] Chris North and Ben Shneiderman. Snap-together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, pp. 128–135, 2000.

[PDF14] Charles Perin, Pierre Dragicevic, and Jean-Daniel Fekete. Revisiting Bertin Matrices: New Interactions for Crafting Tabular Visualizations . *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, No. 12, pp. 2082–2091, 2014.

[PJ95] Steven G. Parker and Christopher R. Johnson. SCIRun: A Scientific Programming Environment for Computational Steering. In *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, Supercomputing '95, 1995.

[Pro] Processing.org. https://processing.org/.

[Qua] Quartz Composer. https://developer.apple.com/library/mac/documentation/GraphicsImaging/Conceptual/QuartzComposerUserGuide/.

[RHY14] Donghao Ren, Tobias Hollerer, and Xiaoru Yuan. iVisDesigner: Expressive Interactive Design of Information Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, No. 12, pp. 2092–2101, 2014.

[RJ13] Sébastien Rufiange and Michael J.McGuffin. DiffAni: Visualizing Dynamic Graphs with a Hybrid of Difference Maps and Animation. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 19, No. 12, pp. 2556–2565, 2013.

[SH14] Arvind Satyanarayan and Jeffrey Heer. Lyra: An Interactive Visualization Design Environment. *Computer Graphics Forum (Proc. EuroVis)*, Vol. 33, No. 3, pp. 351–360, 2014.

[Sim] Simulink. http://jp.mathworks.com/products/simulink/.

[Siv] Play Siv3D! http://play-siv3d.hateblo.jp/.

[SKL+14] Charles D. Stolper, Minsuk Kahng, Zhiyuan Lin, Florian Foerster, Aakash Goel, John Stasko, and Duen Horng Chau. GLO-STIX: Graph-Level Operations for Specifying Techniques and Interactive eXploration. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, No. 12, pp. 2320–2328, 2014.

[SMT12] Satoko Shiroi, Kazuo Misue, and Jiro Tanaka. ChronoView: Visualization Technique for Many Temporal Data. In *Proceedings of the 2012 16th International Conference on Information Visualisation*, IV'12, pp. 112–117, 2012.

[Squ] Squirrel - The Programming Language. http://squirrel-lang.org/.

[TG02] Masahiro Takatsuka and Mark Gahegan. GeoVISTA Studio: A Codeless Visual Programming Environment For Geoscientific Data Analysis and Visualization. *Computational Geoscience*, Vol. 28, pp. 1131–1144, 2002.

[Tuf90] Edward R. Tufte, editor. *Envisioning Information*. Graphics Pr, 1990.

[UFK+89] Craig Upson, Thomas Faulhaber, Jr., David Kamins, David H. Laidlaw, David Schlegel, Jefrey Vroom, Robert Gurwitz, and Andries van Dam. The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE Comput. Graph. Appl.*, Vol. 9, No. 4, pp. 30–42, 1989.

[vs2] Visual Studio 2015 - Visual Studio. https://www.microsoft.com/ja-jp/dev/products/visual-studio-2015.aspx.

[Wea04] Chris Weaver. Building Highly-Coordinated Visualizations in Improvise. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pp. 159–166, 2004.

[Xta] xtal-language. https://code.google.com/archive/p/xtal-language/.

[ZZD14] Emanuel Zgraggen, Robert C. Zeleznik, and Steven M. Drucker. Panoramicdata: Data analysis through pen & touch. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, No. 12, pp. 2112–2121, 2014.

[伊藤 15] 伊藤隆朗, 三末和男, 田中二郎. データフロービジュアル言語を用いた情報可視化システムの開発環境. 情報処理学会研究報告（第 162 回ヒューマンコンピュータインタラクション研究会）, 2015-HCI-162, 2015.

[伊藤 16] 伊藤隆朗, 三末和男. データフロービジュアル言語を用いた多次元データ可視化手法の開発環境. 情報処理学会論文誌, Vol. 57, No. 7, pp. 1638–1651, 2016.

[出原 86] 出原栄一, 吉田武夫, 渥美浩章（編）. 図の体系 ― 図的思考とその表現. 日科技連, 1986.

[小林 14] 小林愛実. 連結図における重み付きグラフのエッジ表現に関する研究. 筑波大学大学院博士課程 システム情報工学研究科修士論文, 2014.

[石川 14] 石川凜太郎, 三末和男, 田中二郎. グリフ作成を支援するドローツールの開発. 第 13 回情報科学技術フォーラム（FIT2014）, pp. 103–106, 2014.

[椎尾 10] 椎尾一郎（編）. Computer Science Library-11 ヒューマンコンピュータインタラクション入門. サイエンス社, 2010.

[浜中 08] 浜中誠. スクリプト言語による効率的ゲーム開発 C/C++への Lua 組込み実践. ソフトバンククリエイティブ, 2008.

著者論文リスト

本研究に関連する論文

論文誌

[1] 伊藤 隆朗, 三末 和男: データフロービジュアル言語を用いた多次元データ可視化手法の開発環境, 情報処理学会論文誌, Vol. 57, No. 7, pp. 1638-1651, 2016.

査読付き国際会議論文

[2] Takao Ito, Kazuo Misue: A Visualization Technique Using Loop Animations, In *Human Interface and the Management of Information: Information, Design and Interaction - 18th International Conference, HCI International 2016, Proceedings, Part I, LNCS 9734*, pp. 136-147, 2016.

[3] Takao Ito, Kazuo Misue: Development Environment of Embeddable Information-Visualization Methods, In *19th International Conference, HCI International 2017, Proceedings*, 2017. (To Appear)

その他の論文

[4] 伊藤 隆朗, 三末 和男, 田中 二郎: データフロービジュアル言語を用いた情報可視化システムの開発環境，情報処理学会研究報告（第 162 回ヒューマンコンピュータインタラクション研究会）, 2015-HCI-162, 2015 .（**学生奨励賞, 2015 年度山下記念研究賞**）

本研究に関連しない論文

査読付き国際会議論文

[5] Takao Ito, Kazuo Misue, Jiro Tanaka: Drawing Clustered Bipartite Graphs in Multi-Circular Style, In *Proceedings of the 2010 14th International Conference on Information Visualisation*, IV'10, pp. 23-28, 2010.

[6] Takao Ito, Kazuo Misue, Jiro Tanaka: Sphere Anchored Map: A Visualization Technique for Bipartite Graphs in 3D, In *Human-Computer Interaction. Novel Interaction Methods and Techniques - 13th International Conference, HCI International 2009, Proceedings, Part II, LNCS 5611*, pp. 811-820, 2009.

その他の論文

[7] 伊藤 隆朗, 三末 和男, 田中 二郎: タワーマップ： 2 部グラフ構造と量的情報を同時提示する 3 次元可視化手法, 全国大会講演論文集 第 72 回 (インタフェース), pp. 353-354, 2010.

[8] 伊藤 隆朗, 三末 和男, 田中 二郎: 大規模二部グラフにおける球状三次元アンカーマップの描画とインタラクティブな閲覧手法, 全国大会講演論文集 第 70 回 (インタフェース), pp. 317-318, 2008.（学生奨励賞）

ソフトウェア紹介記事

[9] 伊藤 隆朗 : キャラミん: ダンスモーション自動生成を備えた 3D キャラクタモデルを楽しむためのソフトウェア，コンピュータソフトウェア，第 31 巻，第 2 号，pp.38-43, 2014.

ポスター発表

[10] 伊藤 隆朗，三末 和男，田中 二郎: Sphere Anchored Map：大規模 2 部グラフの 3D 描画手法, インタラクティブシステムとソフトウェアに関するワークショップ (WISS 2007), 2 pages, 2007.